

(11)Publication number : 2001-357071
(43)Date of publication of application : 26.12.2001

(72)Inventor : BASS BRIAN MITCHELL
CALVIGNAC JEAN LOUIS
HEDDES MARCO C
ANTONIOS MARAGUKOSU
SIEGEL MICHAEL STEVEN
VERPLANKEN FABRICE JEAN

Priority number : 2000 543531 Priority date : 06.04.2000 Priority country : US

```

graph TD
    1000([スタート]) --> 1001[パルス生成]
    1001 --> 1002[直交テーブルを  
読み出す]
    1002 --> 1003([終了])
    1002 --> 1004{パルス数  
比較}
    1004 -- Yes --> 1005[リレーを  
読み出す]
    1004 -- No --> 1006[PCBを  
読み出す]
    1006 --> 1004
    1005 --> 1007[パルス数  
比較を  
終了]
    1007 --> 1008{パルス数  
比較}
    1008 -- Yes --> 1009([OK終了])
    1008 -- No --> 1010([NG終了])
  
```

Flowchart of the first embodiment of the invention. The process starts at 1000 (Start), proceeds to 1001 (Pulse generation), then to 1002 (Load orthogonal table). From 1002, it branches to 1003 (End) or 1004 (Pulse count comparison). If the comparison at 1004 is "Yes", it proceeds to 1005 (Load relay), then to 1007 (End pulse count comparison), and then to 1008 (Pulse count comparison). If the comparison at 1004 is "No", it proceeds to 1006 (Load PCB), then loops back to 1004. From 1008, if the comparison is "Yes", it proceeds to 1009 (OK End), and if "No", it proceeds to 1010 (NG End).

BEST AVAILABLE COPY

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2001-357071
(P2001-357071A)

(43) 公開日 平成13年12月26日 (2001. 12. 26)

(51) Int.Cl. ⁷	識別記号	F I	テマコード [*] (参考)
G 0 6 F 17/30	4 1 2 3 5 0 4 1 4	G 0 6 F 17/30	4 1 2 3 5 0 A 4 1 4 Z
H 0 4 L 12/46		H 0 4 L 12/46	A

審査請求 有 請求項の数34 OL (全 25 頁)

(21) 出願番号 特願2001-91839(P2001-91839)
(22) 出願日 平成13年3月28日 (2001. 3. 28)
(31) 優先権主張番号 09/543531
(32) 優先日 平成12年4月6日 (2000. 4. 6)
(33) 優先権主張国 米国 (US)

(71) 出願人 390009531
インターナショナル・ビジネス・マシー
ズ・コーポレーション
INTERNATIONAL BUSIN
ESS MACHINES CORPO
RATION
アメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)
(74) 代理人 100086243
弁理士 坂口 博 (外2名)

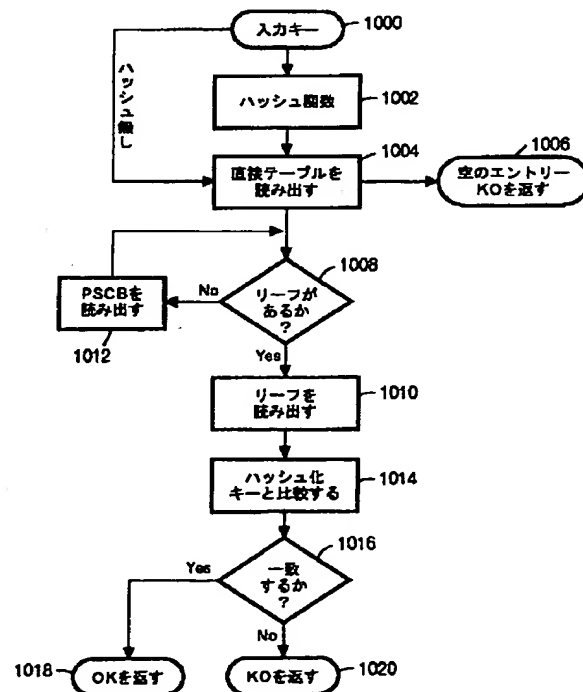
最終頁に続く

(54) 【発明の名称】 ネットワーク・プロセッサ用の完全一致 (FM) サーチ・アルゴリズムの実装

(57) 【要約】 (修正有)

【課題】 サーチ・パターンとサーチ・ツリーのリーフに格納されたパターンとの間の完全一致を見つけるための新規のデータ構造、方法、および装置を提供する。

【解決手段】 キーを入力し、キーに対しハッシュ関数を実行し、直接テーブル (DT) にアクセスし、パターン・サーチ制御ブロック (PSCB) を通してリーフに達するまでツリーのウォーク・スルーを行う。キーおよび検索のために必要な対応する情報の両方が、パトリシア・ツリー構造に格納され、ハッシュ関数は、キーのビットからハッシュ・キーのビットへ $n \rightarrow n$ のマッピングを行う。



【特許請求の範囲】

【請求項1】入力キーをサーチ文字列として読み出す動作と、

ハッシュ関数を使用して前記入力キーをハッシュ化してハッシュ化キーを生成する動作と、

前記ハッシュ化キーの最上位Nビットを、各々の非空エントリがサーチ・ツリーの次の分岐またはリーフのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用する動作と、

非空テーブル・エントリのポインタが、対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決定する動作と、

ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐内容を読み出す動作と、

対応するサーチ・ツリーのリーフに達した場合、リーフ内容を読み出し、リーフのパターンを前記ハッシュ化キーと比較して、前記リーフ・パターンが前記ハッシュ化キーと一致するかどうかを決定する動作と、

前記リーフ・パターンが前記ハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返す動作とを含む、コンピュータ処理装置によって可変長サーチ・キーの完全一致を決定するための方法。

【請求項2】前記サーチ・ツリーの複数のルート・ノードを表すテーブルが2^N個のエントリを含む、請求項1に記載の完全一致を決定するための方法。

【請求項3】前記コンピュータ処理装置がネットワーク・プロセッサである、請求項1に記載の完全一致を決定するための方法。

【請求項4】前記対応するサーチ・ツリーの次の分岐の内容が別の次の分岐を指し示す、請求項1に記載の完全一致を決定するための方法。

【請求項5】前記次の分岐の内容が対応するサーチ・ツリーのリーフを指し示す、請求項1に記載の完全一致を決定するための方法。

【請求項6】前記リーフ・パターンが前記ハッシュ化キーと一致せず、かつ別のリーフのポインタを含まない場合、一致見つからずの標識を返すことをさらに含む、請求項1に記載の完全一致を決定するための方法。

【請求項7】テーブルのインデックスが空エントリのインデックスである場合、一致見つからずの標識を返すことをさらに含む、請求項1に記載の完全一致を決定するための方法。

【請求項8】色レジスタの内容を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、請求項1に記載の完全一致を決定するための方法。

【請求項9】ゼロの文字列を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、請求項1に記載の完全一致を決定するための方法。

【請求項10】前記次の分岐のビット数が前記ハッシュ

化キーの長さを超える場合、完全一致のサーチを終了する動作をさらに含む、請求項1に記載の完全一致を決定するための方法。

【請求項11】前記入力キーに対して使用される前記ハッシュ関数が、前記ハッシュ化キーを前記入力キーに変換できる可逆ハッシュ関数である、請求項1に記載の完全一致を決定するための方法。

【請求項12】前記リーフが別のリーフへのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、

前記格納されたパターンが前記ハッシュ化キーと一致せず、かつ前記チェーンの次のリーフのポインタを含まない場合、一致見つからずの標識を返す動作とをさらに含む、請求項1に記載の完全一致を決定するための方法。

【請求項13】前記リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、

前記格納されたパターンが前記ハッシュ化キーと一致する場合、一致発見の標識を返す動作とをさらに含む、請求項1に記載の完全一致を決定するための方法。

【請求項14】サーチすべきパターンまたはキーと、サーチ・ツリーの第1アドレス位置を格納する直接テーブルと、

各々がサーチ・ツリーの分岐を表す複数のパターン・サーチ制御ブロックと、

各リーフがサーチの結果のためのアドレス位置である複数のリーフとを含む、可変長サーチ・キーの完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項15】ツリー・サーチ・メモリを管理するルックアップ定義テーブルをさらに含む、請求項14に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項16】前記ルックアップ定義テーブルは、ツリーが存在する物理的メモリ、キーおよびリーフのサイズ、および実行すべきサーチの種類を定義するエントリを含む、請求項15に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項17】前記ルックアップ定義テーブルが複数のメモリに実装される、請求項14に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項18】直接テーブル・エントリのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、次の試験ビット、および直接リーフのうちの少なくとも1つを

含む、請求項14に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項19】パターン・サーチ制御ブロックのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、および次の試験ビットのうちの少なくとも1つを含む、請求項14に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項20】リーフ・データ構造が1つのリーフ連鎖ポインタ、プレフィックス長、サーチ・キーと比較すべきパターン、および可変ユーザ・データのうちの少なくとも1つを含む、請求項14に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項21】前記直接リーフは直接テーブル・エントリに直接格納され、かつサーチ制御ブロックおよびサーチ・キーと比較されるパターンを含む、請求項18に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項22】パターン・サーチ制御ブロックが、サーチ・ツリーのリーフ・パターンが異なる位置に挿入される、請求項14に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項23】パターン・サーチ制御ブロックは1の幅および1の高さによって定義される形状を持ち、かつ少なくとも36ビットのライン長を有するメモリに格納される、請求項14に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項24】フレーム処理を行う複数のプロトコル・プロセッサおよび内部制御点プロセッサを含む埋込み式プロセッサ複合体と、

各プロトコル・プロセッサにアクセスでき、高速パターン・サーチ、データ操作、およびフレーム・パージングをもたらす複数のハードウェア・アクセラレータ・コプロセッサと、

少なくとも1つのサーチ・ツリーを表す複数のデータ構造であって、直接テーブルとパターン・サーチ制御ブロックとリーフとを含む前記データ構造を格納する複数のプログラム可能なメモリ装置と、

各プロトコル・プロセッサの複数のメモリ装置へのアクセスを制御する制御メモリ・アービタを含む、可変長サーチ・キーの完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項25】プロトコル・プロセッサの実行と並行して作動して、メモリの読み書きおよびメモリ範囲検査を含むツリー・サーチ命令を実行するツリー・サーチ・エンジンを含み、請求項24に記載の完全一致を決

定するために半導体基板上に組み立てられた装置。

【請求項26】前記複数のメモリ装置は内部スタティック・ランダム・アクセス・メモリ、外部スタティック・ランダム・アクセス・メモリ、および外部ダイナミック・ランダム・アクセス・メモリの少なくとも1つをさらに含む、請求項24に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項27】前記制御メモリ・アービタが、複数のプロトコル・プロセッサと複数のメモリ装置との間でメモリ・サイクルを割り当てることによって、制御メモリ動作を管理する、請求項24に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項28】各プロトコル・プロセッサは一次データ・バッファ、スクラッチ・パッド・データ・バッファ、およびデータ格納動作のための制御レジスタを含む、請求項24に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項29】サーチ・キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに含む、請求項24に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項30】プログラム可能なサーチ・キー・レジスタおよびプログラム可能なハッシュ化キー・レジスタをさらに含む、請求項24に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項31】複数の独立サーチ・ツリーの間で単一のテーブル・データ構造を共用することを可能にするプログラム可能な色キー・レジスタをさらに含む、請求項30に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項32】色レジスタがイネーブルされている場合、その内容をハッシュ出力に付加して最終ハッシュ化キーを生成する、請求項31に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項33】色レジスタがイネーブルされていない場合、同等の数のゼロをハッシュ出力に付加して最終ハッシュ化キーを生成する、請求項31に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項34】入力キーをサーチ文字列として読み出すプログラム命令と、

ハッシュ化キーを入力キーに変換することができる可逆ハッシュ関数を使用して前記入力キーをハッシュ化するプログラム命令と、

前記ハッシュ化キーの最上位Nビットを、各非空エントリがサーチ・ツリーの次の分岐またはリーフへのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用するプログラム命令と、

非空テーブル・エントリのポインタが対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決

10

20

30

40

50

定するプログラム命令と、

ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐の内容を読み出すプログラム命令と、

対応するサーチ・ツリーのリーフに達したときにリーフ内容を読み出し、リーフのパターンをハッシュ化キーと比較して、リーフ・パターンがハッシュ化キーと一致するか否かを決定するプログラム命令と、

リーフ・パターンがハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返すプログラム命令とを含む、可変長サーチ・キーの完全一致を決定するためのコンピュータ・プログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般的に、パターン・マッチング・アルゴリズムに関し、さらに詳しくは、ネットワーク処理装置に実装することができる完全マッチング・サーチ・アルゴリズムに関する。

【0002】

【従来の技術】ますます複雑になるタスクをメディア速度でサポートするためのハードウェア統合処理の要求から、ネットワーク・プロセッサが生成された。ネットワーク・プロセッサは、1組の埋込み型プログラム可能プロトコル・プロセッサおよび相補型システム・コプロセッサにより機能柔軟性を備えた配線速度のフレーム処理および転送能力を提供する。ネットワーク・プロセッサは、今日のパーソナル・コンピュータにとってのマイクロプロセッサと同様に、ネットワークにとっての基本的ネットワーク構築ブロックになるものと期待される。ネットワーク・プロセッサは複数のデータ・ストリームの実時間処理を提供し、セキュリティの強化やIPパケット処理および転送能力をもたらす。さらに、それは、並列分散処理およびパイプライン処理設計などの高度アーキテクチャにより、速度改善をもたらす。これらの能力は効率的なサーチ・エンジンを可能にし、データ処理スループットを向上し、複雑なタスクの高速実行をもたらす。ネットワーク・プロセッサのプログラム可能な機能は、ネットワーク製品開発者に、新しい特注のアプリケーション特定の集積回路(ASIC)の設計を必要とすることなく、新しいプロトコルおよび技術を実装するためのより容易な移行経路を提供する。

【0003】ネットワーク・プロセッサは、インターネット・プロバイダまたはエンタープライズ・ネットワーク・プロバイダ向けの相互接続解決策を開発するための高度にカスタマイズ可能でスケーラブルな技術を提供する。ネットワーク・プロセッサは、ローエンド独立型装置から大型マルチラック解決策まで広範囲の解決策の基礎を提供する。この性質の規模変更は、高性能非ブロッキング・パケット・ルーティング・スイッチ技術、および他の産業スイッチ技術に適応できるIBM社のData A

igned Serial Link(DASL)インタフェースなど専有技術のインタフェースを使用することにより達成される。

【0004】プログラム可能な通信集積回路として、ネットワーク・プロセッサは、非常に効率的なパケット分類、フレーム毎のマルチテーブル・ルックアップ、パケット変更、キュー/ポリシー管理、およびその他のパケット処理能力を提供する。ネットワーク・プロセッサは、スイッチング・エンジン、サーチ・エンジン、フレーム・プロセッサ、およびイーサネット(登録商標)MACを1つの装置に統合して、いずれかのプロトコル層でフレーム内容に基づき高容量メディア重みスイッチング・フレームを必要とする顧客の要求をサポートする。

【0005】ハードウェア・アクセラレータは、フレーム転送、フレーム・フィルタリング、およびフレーム変更を実行する。複合範囲および動作仕様を持つ何百ものルールを実施するネットワーク・プロセッサの能力は、フィルタリング能力の新しいベンチマークを設定し、ネットワーク・プロセッサに基づくシステムを高容量サーバ・ファーム・アプリケーションに独特に適応させる。

【0006】ネットワーク・プロセッサを用いて展開される典型的なシステムは、分散型ソフトウェア・モデルを使用し、各プログラム可能なネットワーク・プロセッサがタスクを同時に実行する。幾つかの機能は制御点(CP)プロセッサで実行され、これはネットワーク・プロセッサの内部または外部に置くことができる。CPは、レイヤ2およびレイヤ3のルーティング・プロトコル、ならびにレイヤ4およびレイヤ5のネットワーク・アプリケーションおよびシステム管理をサポートする。配線速度の転送およびフィルタリング機能は、ネットワーク・プロセッサ・ハードウェアおよび常駐ピココードの組合せによって実行される。

【0007】多数の相互接続されたノードを含む通信ネットワークで、データは1つのノードから他のいずれかのノードまたはネットワークに送信することができる。ルータと呼ばれる特殊化されたノードは、データをその宛先に転送する能力を持つ。通信ネットワークを通して送信されるデータは、一般的にヘッダの一部として、宛先アドレスに関する情報を含む。各ルータはこの情報または少なくともその一部を、内部に格納されたアドレスのリストと比較する。格納されたアドレスと宛先アドレスとの間に一致が見つくと、ルータはその宛先ノードに通じる経路を確立する。ネットワークの規模および構造によって、データはその宛先に直接転送されるか、または他の中間ルータへ送信される。国際標準化機構(ISO)は、ルータが部分アドレスのルーティング情報を格納するルーティング規格を発表した。ルータはそこで、そのデータベースにある最もよく一致する部分アドレスにパケットを送信する。ISO規格は、所定の桁数または所定のヘッダ長を使用してノードの階層構造を構

築することを可能にする。主ルータはアドレスの最初の部分によってアドレス指定され、サブルータは中間部分によって、かつ最終宛先はアドレスの最後の数桁によって指定される。したがって、いずれかのルータが、データの送信先の階層のレベルに割り当てられた桁を読み取れば充分である。

【0008】受信パケットのルーティングは、付随するアドレス文字列に基づく。アドレス文字列は、アドレス文字列と共に、どのルータがパケットの配信における次のルータであるかなど他の関連詳細を含むデータベースで、サーチ・キーとして使用される。データベースはルーティング・テーブルと呼ばれ、現在のルータと次のルータとの間のリンクは、パケットの進行における次のホップと呼ばれる。ルーティング・テーブル・サーチ・プロセスは、アドレスの構造のみならず、テーブルの編成にも依存する。たとえば非階層構造を有する7ビット以下のサイズのサーチ・キーは、一連のアドレス・エントリとして編成されたルーティング・テーブルでは最も効率的に見つかる。サーチ・キーは、テーブルで正しいエントリをつき止めるための索引として使用される。より大きなサイズ、たとえば32ビットのサーチ・キーの場合、対応するルーティング・テーブルは、10,000個を超えるエントリを持つことができる。データベースを索引によって直接サーチする単純なテーブルとして編成すると、テーブルの大部分が空になるので、大量のメモリ空間を無駄にすることになる。

【0009】従来のルータは、サーチ・プロセスを幾つかのステップに分割する。第1ステップは、ルータが宛先ホスト・コンピュータに直接接続されているかどうかを決定することである。この場合、メッセージは宛先から1ホップであり、その方向に経路選択しなければならない。宛先コンピュータがそのルータに直接接続されていない場合、次のステップは、宛先ネットワークのトポロジカル方向を決定することである。トポロジカル・レイアウトから方向が決定されると、メッセージはその方向に経路選択される。そうでない場合、最終ステップは、メッセージをデフォルト・リンクに沿って経路選択することである。

【0010】一般的に、第1ステップは、ルータに直接接続された32ビットのアドレスのホスト・コンピュータを含むテーブル内の線形サーチを使用して実行される。ローカル・トポロジを反映して、アドレス・テーブル内の各エントリは、アドレス指定されたコンピュータに直接通じている対応する出力インタフェースに接続される。宛先アドレスをルータが受け取ると、32ビット全部が、テーブル内の宛先アドレスの各々と比較される。一致が見つかると、メッセージは、指定されたルータ・インタフェースを介して、対応する宛先に直接送信される。

【0011】第2ステップ、つまり宛先ネットワークの

方向を決定するステップは通常、ネットワーク・アドレスの数のためにそのようなテーブルの管理および使用が困難になるので、テーブル内の線形サーチによって行われることはない。従来の技術では、アドレス文字列がネットワーク・アドレス、サブネット・アドレス、およびホスト識別の3つの階層レベルに一致すると、ルータは、ハッシュ化、パトリシア・ツリー・サーチ、およびマルチレベル・サーチなど、幾つかの周知の技術の1つを使用して決定を行った。ハッシュ化で、ハッシュ関数はアドレスのネットワーク部分を縮小し、小さい管理可能な索引を生成する。ハッシュ索引は、ハッシュ・テーブルに索引を付け、一致するハッシュ・エントリを探索するために使用される。ハッシュ・テーブルの各ハッシュ・エントリに対応するのが、対応するネットワークのトポロジカル方向を指し示す出力インタフェースのアドレスである。ハッシュ・ネットワーク部分とハッシュ・エントリとの間に一致が見つかると、メッセージは、対応するインタフェースおよび宛先ネットワークの方向に向けられる。

【0012】ハッシュ化により、大きい管理不能なフィールドは、小さい管理可能な索引に縮小される。しかし、このプロセスでは、2つまたはそれ以上のフィールドから同一ハッシュ索引が生成されることがある。これらのフィールドはハッシュ・テーブルの同一場所に格納しなければならないので、このできごととは衝突と呼ばれる。衝突中にエントリを区別するために、さらなるサーチが必要である。したがって、衝突は、ハッシュ・サーチを用いることから得られる効率を低下し、全ての許容アドレスが単一の索引に縮小される最悪の場合には、ハッシュ化はサーチ・プロセスとして事実上役立たないとみなされる。

【0013】パトリシア・ツリー・サーチは、ハッシュ法によって生じる衝突を回避する。このサーチ法は、全てのアドレス文字列およびたとえば関連ルート情報などの付随情報をバイナリ・ツリーに格納する必要がある。このサーチ・プロセスは、アドレスをアドレス文字列内の最上位ビット位置から1ビットずつ、ツリー・ノードと比較する。一致したビット値は、左または右子ノードのいずれかに進むようにサーチを誘導し、このプロセスがアドレスの次のビットに対して繰り返される。サーチ時間は、格納された最長アドレス文字列のサイズに比例する。パトリシア・ツリー・サーチでは、平均サーチ時間と最悪の場合のサーチ時間との間の差は、あまり大きくない。さらに、ルーティング・テーブルは極めて効率的に編成される。それは、ハッシュ法の匹敵するルーティング・テーブルより少ないメモリを必要とする。パトリシア・ツリー・サーチは、最悪の場合のサーチをハッシュ法よりよく処理するが、大抵の場合、一致をつき止めるまでの時間がかなり長くなる。したがって、多くの従来のルータは、ハッシュ化とパトリシア・ツリー・

サーチの組合せを使用する。この組合せはマルチレベル・サーチと呼ばれる。

【0014】マルチレベル・サーチは、ハッシュ化とパトリシア・ツリー・サーチを結合したものである。キャッシュは、最も最近に、かつおそらく最も一般的に経路選択されたネットワーク・アドレスの部分セットを含むハッシュ・テーブルを格納し、パトリシア・ツリーはネットワーク・アドレスの全セットを格納する。メッセージを受信すると、宛先アドレスはテーブル上にハッシュ化される。それが予め定められた時間内につき止められない場合、アドレスは、アドレスが格納されていれば見つかることを確実にするパトリシア・ツリー・サーチ・エンジンに渡される。

【0015】従来の技術には、固定マッチ・ツリー、最長プレフィックス・マッチ・ツリー、およびソフトウェア管理ツリーをはじめ、多数の既知のツリー・サーチ・アルゴリズムがある。固定マッチ・ツリーは、レイヤ2イーサネットMACテーブルなど、厳密な一致を要求する固定サイズ・パターンに使用される。最長プレフィックス・マッチ・ツリーは、IPサブネット転送など、部分一致だけを必要とする可変長パターンに使用される。ソフトウェア管理ツリーは、フィルタ・ルールなど範囲またはビット・マスクとして定義されるパターンに使用される。一般的に、ルックアップはツリー・サーチ・エンジン(TSE)を用いて実行される。

【0016】

【発明が解決しようとする課題】本発明の目的は、パトリシア・ツリーのための完全一致ツリー・サーチ・アルゴリズムをハードウェア内に実装することである。このアルゴリズムの目的を果たすことができるようにメモリ構造をどのようにセットアップし、これらの構造をハードウェアがどのように処理するかを記載する。

【0017】本発明の別の目的は、以前のポイントに関する記憶を必要とせず、試験する次のビットまたはビット群と共に正方向ポイントのみを使用し、それによりノードの記憶空間を縮小するサーチ機構を提供することである。

【0018】

【課題を解決するための手段】主要な概念は、キーを入力し、キーに対してハッシュ関数を実行し、直接テーブル(DT)にアクセスし、パターン・サーチ制御ブロック(PSCB)を通してツリーのウォーク・スルーを実行して、リーフで終了するというものである。

【0019】解決される問題は、少数のレジスタおよび正規メモリに配置することができ、次いで比較的単純なハードウェア・マクロによって操作できるパトリシア・ツリー構造を構築するために使用できる、1組のデータ構造の設計である。パトリシア・ツリーに、キーおよび検索のために必要な対応する情報の両方が格納される。

【0020】キーは、探索され照合される情報である。

最初に、キーがレジスタに入れられ、ハッシュ化される。その結果がハッシュ・キーであり、実際のサーチはハッシュ・キーに対して行われる。ハッシュ関数は空ハッシュとすることができ、その場合ハッシュ・キーはキーと全く同一となる。ハッシュ関数は、キーのビットからハッシュ・キーのビットへ $n \rightarrow n$ のマッピングを行う。

【0021】ハッシュ・キーおよび関連情報をツリーに格納するために使用されるデータ構造は、リーフと呼ばれる。リーフを検索することが、このアルゴリズムの目的である。各リーフは、入力キーと厳密に一致する単一のキーに対応する。この実現ではリーフはキーを含み、格納すべき追加情報がそれに付加される。リーフの長さは、キーの長さの場合と同様に、プログラム可能である。リーフはランダム・アクセス・メモリに格納され、単一メモリ・エントリとして実現される。キーが直接テーブル(DT)内にある場合には、それは直接リーフと呼ばれる。

【0022】

【発明の実施の形態】本発明について、本発明が埋め込まれたネットワーク・プロセッサの文脈で説明する。ネットワーク・プロセッサ10は、単一チップ上のプログラム可能なスイッチングおよびルーティング・システムであり、そのアーキテクチャを図1に示す。それは、スイッチ・インタフェースに接続するためのData Aligned Serial Link(DASL)のみならず、10/100イーサネット、ギガビット・イーサネット、およびPacket over SONET(POS)用のメディア・インタフェースを提供する。内部ハードウェア・アクセラレータは、性能および効率を高める。埋込み型プロセッサ複合体(EPC)12は、プロトコル・プロセッサおよびフレーム処理、構成、および管理サポートのための内部制御点プロセッサを含む。

【0023】最高N個までの並列プロトコル・プロセッサが利用可能である。16個のプロトコル・プロセッサの実施形態の場合、16、384語の内部ビココード命令記憶装置および32、768語の外部ビココード命令記憶装置が利用可能である。毎秒212、800万個の命令(MIPS)の集合処理能力を提供することができる。さらに、各プロトコル・プロセッサは、高速パターン・サーチ、データ操作、内部チップ管理機能、フレーム・パージング、およびデータ先取りサポートを提供する。M個のハードウェア・アクセラレータ・コプロセッサにアクセスできる。好ましい実施形態では、プロトコル・プロセッサ用の制御記憶装置は、内部および外部メモリの両方、すなわち即時アクセス用の32Kの内部スタティック・ランダム・アクセス・メモリ(SRAM)28、高速アクセス用の外部ゼロ・バス・ターンアラウンド(ZBT)SRAM30、および大容量記憶要求用の外部二重データ・レート(DDR)ダイナミック・ラ

ンダム・アクセス・メモリ(DRAM)32によって提供される。

【0024】接続された制御点プロセッサ34で動作する前処理アルゴリズムと共に埋込み型ハードウェア・アクセラレータを使用して、ネットワーク・プロセッサ10は、複合範囲、優先順位、および動作仕様を持つ100またはそれ以上のフィルタ・ルールにより、配線速度でフレームを処理することができる。これにより、ネットワーク・プロセッサに基づくシステムは、ゲートウェイ、サーバ・ファーム・アプリケーション、およびトラフィックの混合の処理に関連するフィルタリング・タスクによく適合するようになる。

【0025】ネットワーク管理者がコヒーレントでユーザ・フレンドリーなインタフェースにフィルタ・ルールを入力すると、制御点ソフトウェアは自動論理検査を行う。安定度論理に基づき新規のフロー制御を使用して、ネットワーク・プロセッサ10は、伝送制御プロトコル(TCP)のコラプス無く、一般的に使用されるランダム早期廃棄法より高速の一時オーバーサブスクリプションに耐える。ネットワーク・プロセッサ10はまた、帯域幅を自動的に割り当てることによって、Differentiated Servicesをも配信し、ネットワーク管理者は、瞬時または仮定トラフィック統計に基づき多数の閾値を設定する効果を予測する必要性から解放される。

【0026】単一ネットワーク・プロセッサ10は、最高40までの高速イーサネット・ポートまたは4つまでのギガビット・イーサネット・ポートのメディア速度スイッチングを行う。それはまた、OC-48c、OC-48、4つのOC-12、または16のOC-3ポートをサポートするように構成することもできる。スケーラビリティのために、2つの3.5GbpsシリアルDASLリンクを使用して、2つのネットワーク・プロセッサを相互接続してポート密度を倍加するか、またはスイッチ構造を接続して最高64までのネットワーク・プロセッサによるスイッチング解を生成することができる。システムのアベイラビリティを高めるため、2つのDASLリンク、つまり1つの一次および1つの二次DASLリンクを、冗長スイッチ構造に接続することもできる。

【0027】ネットワーク・プロセッサ10の1つの例示的实施形態は、図1に示す通り次の主要部分を含む。

1. 最高16までのプログラム可能なプロセッサとコプロセッサを含む埋込み型プロセッサ複合体(EPC)12
2. フレームがイーサネット物理層装置からスイッチ構造へ移動するためのエンキュー・デキュー・スケジューリング論理14(EDSイングレス)
3. フレームがスイッチ構造からイーサネット物理層装置へ移動するためのエンキュー・デキュー・スケジューリング論理16(EDSエグレス)

4. 別のネットワーク・プロセッサまたは中間スイッチへの相互接続のためのイングレス・スイッチ・インタフェース(スイッチ・イングレス)18およびエグレス・スイッチ・インタフェース(スイッチ・エグレス)20のDASLリンク

5. イーサネットまたはPOS物理層装置26からフレームを受信する物理MACマルチプレクサ22(PMMイングレス)、およびイーサネットまたはPOS物理層装置26へフレームを送信する物理MACマルチプレクサ24(PMMエグレス)

【0028】図2は、埋込み型プロセッサ複合体の例示的实施形態を示す。それは16のプロトコル・プロセッサを含み、2128MIPSの処理力を提供する。各プロトコル・プロセッサ40は、3ステージ・パイプライン(取出し、復号、および実行)、汎用レジスタ、専用レジスタ、8命令キャッシュ、専用論理演算装置(ALU)、およびコプロセッサを含み、全て133MHzで動作する。プロトコル・プロセッサのうち2つは特殊化されている。つまり、1つは誘導フレームの処理用であり(誘導フレーム・ハンドラ)、1つは制御メモリ内のルックアップ・データの構築用である(一般ツリー・ハンドラ)。

【0029】図3は、プロトコル・プロセッサの例示的实施形態を示す。各々のプログラム可能なプロトコル・プロセッサ40に関連付けられるコプロセッサは、次の機能を果たす。

1. データ記憶コプロセッサ64は、フレーム・バッファ・メモリ42、44(イングレスおよびエグレス方向)をインタフェースして直接メモリ・アクセス(DMA)機能をもたらす。
2. チェックサム・コプロセッサ62は、ヘッダのチェックサムを計算する。
3. エンキュー・コプロセッサ66は、キー・フレーム・パラメータを含む256ビット作業レジスタへのアクセスを制御する。このコプロセッサは完了ユニット46とインタフェースして、フレームをスイッチおよびターゲット・ポート・キューに入れる。
4. インタフェース・コプロセッサは、デバッグまたは統計情報の収集のため、全てのプロトコル・プロセッサを内部レジスタ、カウンタ、およびメモリにアクセスさせる。
5. 文字列コピー・コプロセッサは、EPC内におけるデータの効率的な移動を可能にする。
6. カウンタ・コプロセッサは、プロトコル・プロセッサ40のためのカウンタの更新を管理する。
7. ポリシー・コプロセッサはフロー制御情報を検討し、事前に割り当てられた帯域幅と一致するか検査する。

【0030】ハードウェア・アクセラレータ48は、フレーム転送、フレーム・フィルタリング、フレーム変

更、およびツリー・サーチを実行する。ネットワーク・プロセッサに組み込まれるその他の機能として、革新的フィルタ・ルール処理、ハッシュ関数、およびフロー制御が含まれる。

【0031】プロトコル・プロセッサ40は、複合範囲および動作仕様を持つ100またはそれ以上のフレーム・フィルタ・ルールを実施することができる。フィルタリングはネットワーク・セキュリティのために不可欠であり、ネットワーク・プロセッサ・ハードウェア補助機構48は、これらの複合ルール・セットを配線速度で実施する。フィルタ・ルールは、IPヘッダ情報に基づき、フレームを拒絶または許可し、あるいはサービスの品質(QoS)を割り当てることができる。ルールを前処理するための制御点ソフトウェアは、論理エラーを自動的に訂正する。論理的に正しいルール・セットが入力された後、パケット・ヘッダ情報からキーが形成され、ネットワーク・プロセッサのソフトウェア管理されるツリーを用いて、配線速度で試験される。

【0032】幾何学的ハッシュ関数はIPヘッダの統計的構造を利用して、理想的ランダム・ハッシュより優れた性能を発揮する。したがって、低衝突率は、追加的解決サーチ無しで、完全一致テーブルの高速ルックアップを可能にする。

【0033】プロトコル・プロセッサの実行と並行して作動して、ツリー・サーチ・エンジン70はツリー・サーチ命令(メモリ読出し、書込み、または読み書き)、メモリ範囲検査、および不正メモリ・アクセス通知を実行する。図14は、ツリー・サーチ・エンジンの例示の実施形態を示す。

【0034】ネットワーク・プロセッサ10内で2つのシステム制御オプションが利用可能である。内部プロセッサ34は、システムの制御点(CP)プロセッサとして機能することができ、または代替的に、初期化および構成のために4つのイーサネット・マクロの1つに外部プロセッサ接続することができる。CPプロセッサ34は、誘導フレームと呼ばれる特殊なイーサネット・フレームを構築することによって、ネットワーク・プロセッサ内の他のプロセッサ・エンティティと通信する。誘導フレームはDASLリンクをまたいで他の装置へ転送することができ、単一のイーサネット・ポートに接続された1つのCPプロセッサが、サブシステム内に含まれる全てのネットワーク・プロセッサ装置と通信してそれらを制御することを可能にする。各ネットワーク・プロセッサ10の内部プロセッサ34はまた、別個の32ビットPCIバスを用いて通信することもできる。

【0035】ネットワーク・プロセッサ10は通常サブシステム・ボード上にあり、プロトコル層(つまりレイヤ2、レイヤ3、レイヤ4およびより高いレイヤ)のフレーム処理を行う。CPサブシステム内のCPプロセッサ34上で作動するソフトウェアは、管理および経路発

見機能を提供する。CPコード、プロトコル・プロセッサ上で作動するピココード、および誘導フレーム・ハンドラ上で作動するピココードは、このシステムの初期化、転送経路の維持、およびシステムの管理を可能にする。分散システムとして、CPおよび各ネットワーク・プロセッサ・サブシステムは、効率および性能の向上のため、誘導フレームを使用して並行して作動したり通信する複数のプロセッサを含む。

【0036】データ・フレームはPMM22によってメディアから受信され、データ記憶バッファ42に転送される。PMMはまた、受信プロセス中にCRC検査およびフレーム妥当性検証をも実行する。ディスパッチャ50は、フレーム・ルックアップのために利用可能なプロトコル・プロセッサ40に最高64バイトまでのフレーム情報を送信する。分類ハードウェア補助機構48は制御データを供給して、フレーム・フォーマットを識別する。プロトコル・プロセッサ40は制御データを使用して、固定マッチ・ツリー、最長プレフィックス・マッチ・ツリー、またはソフトウェア管理ツリーをはじめ、適用すべきツリー・サーチ・アルゴリズムを決定する。

【0037】ルックアップは、ツリー・サーチ・エンジン(TSE)70を用いて実行される。TSE70は制御メモリ72へのアクセスを実行し、プロトコル・プロセッサ40が実行を続けることを可能にする。制御メモリ72は全てのテーブル、カウンタ、およびピココードによって必要とされるその他のデータを記憶する。効率のために、制御メモリ・アービタ52は、プロトコル・プロセッサ40と様々なオンチップおよびオフチップ制御メモリ・オプション54との間でメモリ・サイクルを割り当てることによって、制御メモリ動作を管理する。

【0038】プロトコル・プロセッサ40は、データ記憶動作のために、一次データ・バッファ、スクラッチ・パッド・データ・バッファ、および制御レジスタ(集合的に72)を含む。ひとたび一致が見つかったら、VLANヘッダ挿入またはオーバーレイなど、イングレス・フレーム変更を適用することができる。これらの変更は、EPC12によっては実行されない。代わりに、ハードウェア・フラグがセットされた場合に、イングレス・スイッチ・インタフェース・ハードウェア18がこの変更を実行する。他のフレーム変更は、ピココードおよびデータ記憶コプロセッサ64により、イングレス・データ記憶装置42内に保持されたフレーム内容を変更することによって、達成することができる。

【0039】フレームをスイッチ構造に送る前に、スイッチ・ヘッダおよびフレーム・ヘッダを構築するために、制御データが収集され使用される。制御データは、フレームの宛先などのスイッチ情報のみならず、エグレス・ネットワーク・プロセッサのための情報を含み、それが宛先ポートのフレーム・ルックアップ、マルチキャストまたはユニキャスト・オペレーション、およびエ

グレス・フレーム変更を促進するのを助ける。

【0040】図4は、例示的なイングレスおよびエグレス・フレームの流れを示す。完了後、エンキュー・コプロセッサ66は、フレームをキュー制御ブロック(QCB)74に入れるために必要なフォーマットを作成し、それらを完了ユニット46に転送する。完了ユニット46は、最高16のプロトコル・プロセッサ40からスイッチ構造キュー76へのフレームの順序を保証する。スイッチ構造キュー76からのフレームは、それらがスイッチ構造76によって転送されるときに、スイッチ・ヘッダおよびフレーム・ヘッダ・バイトを挿入して、64バイト・セルに区分化される。

【0041】スイッチ構造76から受信したフレームは、リアセンブリ制御ブロック(RCB)80およびEDSエグレス44によって提供される情報を使用して、エグレス・データ記憶バッファ78に入れられ、EPC12に登録される。フレームの一部分は、フレーム・ルックアップを実行するために、ディスパッチャ50によってアイドル状態のプロトコル・プロセッサ40に送られる。フレーム・データは、分類ハードウェア補助機構48からのデータと共に、プロトコル・プロセッサ40にディスパッチされる。分類ハードウェア補助機構48は、イングレス・ネットワーク・プロセッサによって形成されたフレーム制御データを使用して、エグレス処理のための開始命令アドレスを決定するのに役立つ。

【0042】エグレス・ツリー・サーチは、イングレス・サーチのためにサポートされるのと同じアルゴリズムをサポートする。ルックアップはTSE70で行われ、プロトコル・プロセッサ40は自由に実行を続けることができる。全ての制御メモリの動作は、プロセッサ複合体の間でメモリ・アクセスを割り当てる制御メモリ・アービタ52によって管理される。

【0043】エグレス・フレーム・データは、データ記憶コプロセッサ64を通してアクセスされる。成功したルックアップの結果は、転送情報、および場合によってはフレーム変更情報を含む。エグレス・フレーム変更は、VLANヘッダの削除、活動時間の増分(IPX)または減分(IP)、IPヘッダ・チェックサムの再計算、イーサネット・フレームCRCオーバーレイ、およびMAC宛先アドレスまたはソース・アドレスのオーバーレイまたは挿入を含むことができる。IPヘッダ・チェックサムは、チェックサム・コプロセッサ62によって作成される。変更が埋込み型プロセッサ複合体12によって実行されるのではなく、ハードウェア・フラグが形成され、PMMエグレス・ハードウェア24が変更を実行する。完了後、フレームをEDSエグレス・キュー44に登録するために、エンキュー・コプロセッサ46を使用して必要なフォーマットが構築され、それらが完了ユニット46に転送される。完了ユニット46は、最高16のプロトコル・プロセッサから、エグレス・イーサ

ネットMACに供給するEDSエグレス・キュー44へのフレーム順序を保証する。完成したフレームは最終的に、PMMエグレス・ハードウェア24によってイーサネットMACまたはPOSインタフェースに送られ、物理ポートから送り出される。

【0044】図14に示すように、ツリー・サーチ・エンジン(TSE)70は、ツリー概念を使用して情報を格納したり検索する。検索つまりツリー・サーチのみならず、挿入や削除も、たとえばMACソース・アドレス、またはIPソース・アドレスとIP宛先アドレスの連結などのビット・パターンであるキーに基づいて行われる。本発明で使用するための例示的ツリー・データ構造100を図5に示す。情報は、少なくともキー102を含むリーフ116、118、120、122と呼ばれる制御ブロックに格納される(格納されたビット・パターンは実際にはハッシュ化キーである)。リーフはエージング情報、またはユーザ情報などの追加情報をも含むことができ、それは、目標ブレードまたは目標ポート番号などの転送情報とすることができる。リーフのフォーマットはピココードによって定義され、オブジェクトは内部または外部制御記憶装置に入れられる。

【0045】ツリーのサーチ・アルゴリズムは、キー102を含む入力パラメータに作用し、キーにハッシュ104を実行し、直接テーブル(DT)108にアクセスし、パターン・サーチ制御ブロック(PSCB)110、112、114を通してツリーのウォーク・スルーを実行し、リーフ116、118、120、122で終了する。各種のツリーはそれ自体のサーチ・アルゴリズムを持ち、異なるルールに従ってツリーのウォークを行わせる。たとえば固定マッチ(FM)ツリーの場合、データ構造はパトリシア・ツリーである。リーフが見つかる、このリーフは、入力キー102と一致することができる唯一の可能な候補である。「最後の比較」演算で、入力キー102はリーフに格納されたパターンと比較される。これにより、リーフが本当に入力キー102と一致するかどうか確認される。このサーチの結果は、リーフが見つかり、かつ一致が発生した場合には成功(OK)であり、その他の場合は全て失敗(KO)である。

【0046】サーチ演算の入力は、次のパラメータを含む。

キー サーチまたは挿入/削除の前に、専用ピココード命令を使用して、176ビット・キーを構築しなければならない。キー・レジスタは1つしか無い。しかし、ツリー・サーチが始動した後、ピココードによってキー・レジスタを使用して、TSE70でサーチを実行すると同時に次のサーチのためのキーを構築することができる。これは、TSE70がキーをハッシュ化し、結果を内部192ビットHashedKeyレジスタ106に格納するからである。

キー長 この8ビット・レジスタは、キー長から1ビットを引いたものを含む。それは、キーの構築中に、ハードウェアによって自動的に更新される。

LUDefIndex これは、サーチが行われるツリーの完全な定義を含むルックアップ定義テーブル (LUDefTable) 内の8ビット・インデックスである。LUDefTableの内部構造を図11に示す。

TSRNr サーチ結果は、1ビット・ツリー・サーチ結果領域TSR0またはTSR1に格納することができる。これは、TSRNrによって指定される。TSE

色がイネーブルされた (LUDefTableで指定される) ツリーの場合、ハッシュ演算中に、16ビット色レジスタ124の内容がキーに挿入される。

【0047】FMツリーの場合、入力キーは、図4に示すようにHashedKey106にハッシュ化される。利用可能な幾つかの固定アルゴリズムがある。使用されるアルゴリズムは、LUDefTableで指定される。

【0048】ルックアップ定義テーブルは、ツリー・サーチ・メモリを管理する主要構造である。LUDefTableは内部メモリ構造であり、ツリーを形成するために128のエントリを含む。LUDefTableは、ツリーが存在する物理メモリ (たとえばDRAM、SRAM、内部RAM)、キャッシングがイネーブルされるかどうか、キーおよびリーフのサイズ、ならびに実行すべきサーチ動作の種類を定義するエントリを含む。LUDefTableは、3つの別個のランダム・アクセス・メモリ、つまり一般プロセッサ・ツリー・ハンドラ (GTH) によってのみアクセス可能な1つのRAM、および相互の複製であり全てのピコプロセッサによってアクセス可能な2つのRAMとして実装される。

【0049】ハッシュ関数104の出力は常に176ビットの数字であり、元の入力キー102とハッシュ関数104の出力との間に1対1の対応があるという特性を有する。以下で説明するように、この特性により、直接テーブル108の後に始まるツリーの深さは最小になる。

【0050】図4の例の場合のように、ツリーに色がイネーブルされている場合、16ビットの色レジスタ124が176ビットのハッシュ関数出力に挿入され、ファイル結果はHashedKey106と呼ばれる192ビットの数字となる。挿入は、直接テーブル108の直後に行われる。直接テーブル108に2ⁿ個のエントリが含まれる場合には、図4に示すように、16ビットの色値がビット位置Nに挿入される。ハッシュ関数の出力は、挿入された色値と一緒に、HashedKeyレジスタ106に格納される。ツリーに対して色がディセーブルされている場合、176ビットのハッシュ関数に変更されずに使用さ

れ、ハッシュ出力に16個のゼロが付加され、192ビットの最終HashedKeyが生成される。

【0051】色は、複数の独立ツリーの間で単一の直接テーブル108を共用するために使用することができる。たとえば色の1つの用途は、MACソース・アドレス (SA) テーブル内のVLAN IDとすることができる。この場合、入力キー102はMAC SAとなり、色124はVLAN IDとなる (VLAN IDは12ビットなので、色の4ビットは使用されず、つまりゼロに設定される)。ハッシュ関数104の後、使用されるパターンは48+16=64ビットである。色は今やパターンの一部であり、異なるVLANのMACアドレスを区別する。

【0052】ハッシュ関数104は、その出力における大部分のエントロピーが最上位ビットに入るように定義される。HashedKeyレジスタ106の最上位のNビットは、直接テーブル (DT) 108内のインデックスを計算するために使用される。

【0053】ツリーを実装する第1構造は、直接テーブル (DT) 108と呼ばれる。N個の要素を持つDTテーブルの各エントリはキーに対応し、その最初のlog₂Nビットは、二進形でDTテーブルのそのエントリのインデックスと同じである。たとえば、16個のエントリのあるDTテーブルにおける5番目のエントリは、最初の3ビットが「0101」であるキーに対応する。最初のlog₂NビットがDTのインデックスと同じであるキーに対応するリーフが無い場合、そのエントリは空と印される。これらのビットに一致するリーフが1つしか無い場合には、そのエントリの中にリーフのポインタがある。このポインタは、そのリーフが格納されているメモリのアドレスである。最初のビットが同一であるキーに対応するリーフが複数ある場合には、DTエントリはPSCB構造110を指し、また次の試験ビット (NBT) フィールド126をも含む。これらの2つの構造について、以下で説明する。

【0054】DTテーブル108はメモリに実装され、そのサイズ (長さ) および開始点はプログラム可能である。別のプログラム可能な特徴は、直接リーフと呼ばれるものを使用することである。DTエントリでリーフを指し示した場合、その後でリーフを読み出さなければならないが、そうする代りにDTエントリの位置にリーフを格納することができる。これは、直接リーフと呼ばれる。これに伴う問題は、言うまでもなく、DTエントリにより多くのメモリを使用することによる速度のトレードオフである。メモリ・サイズ (その幅) はリーフを収容するのに充分でなければならないが、かつ、全てのDTエントリにリーフが格納されるわけではない。しかし、キーの優れたハッシュ関数の結果、リーフのほとんどを単一のDTエントリに付加させることができるので、速度のトレードオフが大きくなることもある。

19

【0055】要するに、DTエントリは空になることがある。この場合、このDTエントリにはリーフが付加されない。DTエントリは、このDTエントリに付加された単一のエントリを指し示すことができる。この場合、DTエントリはパターン・サーチ制御ブロック(PSCB)を指し示すことができ、またそのPSCBの次の試験ビット(NBT)を含むこともできる。このDTエントリに付加されるリーフが2つ以上ある。最後に、DTエントリは直接リーフを含むことができる。

【0056】PSCBは、ツリーの分岐を表す。好ましい実施形態では、0分岐および1分岐がある。PSCBから発する分岐の数は、分岐を指定するために使用されるビット数によって異なる。nビットが使用される場合には、そのPSCBに2ⁿ個の分岐が定義される。各PSCBはまた、ビット位置pにも関連付けられる。PSCBから0分岐を通して到達できる全てのリーフは、パターンの位置pに「0」を持ち、1分岐を通して到達できるリーフは位置pに「1」を持つ。さらに、PSCBから到達することのできる全てのリーフは常に、ビット0・・・p-1が同一のパターンを持つ。つまり、パターンは位置pから異なり始める。PSCBに関連付けられるビット位置は前PSCBまたはDTエントリに格納され、NBT(つまり次の試験ビット)と呼ばれる。PSCBエントリのフォーマットは、DTエントリのフォーマットと同一である。それはランダム・アクセス・メモリに実装される。

【0057】したがって、PSCBはツリーのリーフ・パターンが異なる位置に挿入されるだけである。PSCBの数、およびしたがってサーチ性能は、ツリー内のリーフの数にのみ依存し、パターンの長さには依存しないので、これにより効率的なサーチ演算が可能になる。PSCBレジスタ・フォーマットを図12に示す。

【0058】要するに、PSCBエントリは空になることができ、リーフを指すことができ、または別のPSCBを指すことができ、かつ次の試験ビット(NBT)をPSCBを含むこともできる。FM PSCBは、以下で詳述する通り、1の幅、および1の高さによって定義される形状を持つ。

【0059】PSCBは、2ビット以上に対応する分岐を表すことができる。この場合、たとえば、2ビットに対応するPSCBは、4つのPSCBエントリ、つまり00分岐エントリ、01分岐エントリ、10分岐エントリ、および11分岐エントリを持つことになる。各ツリーは、様々なビット数に対応するPSCBを持つことができる。この場合、前PSCBは、これらのビットが表すビット数のみならず、次のPSCBに対応するビット数をも持つことになる。

【0060】実際の実装では、キーは専用キー・レジスタ102に挿入される。次いでそれはハッシュ化され104、結果はハッシュ化キー・レジスタ106に格納さ

20

れる。ハッシュ関数104はプログラム可能であり、関数の1つは空ハッシュ関数である(つまり、ハッシュ無し)。ハッシュ化キーの最初のnビットは、DTテーブル108のインデックスとして使用される。1つのプログラム可能な機能は、DTエントリのインデックスに使用されるビットの直後にビット・ベクトルを挿入することである。このビット・ベクトルは「色」値(レジスタ124)と呼ばれ、ハッシュ化キーおよび挿入された色値の結果は、ハッシュ化キー・レジスタ106内に格納される。

【0061】FMツリーのリーフのフォーマットは、パターンを含む制御情報を含む。パターンは、リーフをツリー内で一意のものとして識別する。リーフはまた、ツリー・サーチを開始したアプリケーションによって必要とされるデータをも含む。リーフに含まれるデータはアプリケーションによって異なり、そのサイズまたはメモリ要件は、そのツリーのLUDefTableエントリによって定義される。図13は、FMツリーの固定リーフ・フォーマットを示す。

【0062】DTエントリを処理するステップは次の通りである。

- ・DTエントリがメモリから読み出される。
- ・DTエントリが空エントリである場合、これは、最初の「n」ビットがハッシュ化キーと同一であるリーフがツリー内に無いことを意味しているので、サーチは失敗する。
- ・DTエントリにリーフのポインタがある場合には、DT108からのポインタをリーフのアドレスとして使用して、メモリからリーフが読み出される。リーフはレジスタに格納され、キーと比較される。このステップは、最後の比較と呼ばれる。完全一致が存在する場合、ツリー・サーチは成功する。それ以外では、ツリー・サーチは失敗する。

- ・DTエントリがPSCB110のポインタおよびNBTを持つ場合、NBTは最初に特定のレジスタに格納される。次いでNBT数は、キー内のNBT位置のビットを見つけるために使用される。そのビット(0または1)をPSCBのポインタと共に使用して、正確なPSCBエントリが抽出される。つまり、そのビットはポインタの終わりに付加され、PSCBのメモリ内の完全なアドレスを提供する。PSCBは読み出され、特定のレジスタに格納される。次いでハードウェアはPSCBエントリを処理する。この時点で、アルゴリズムはツリーを下方に辿るウォークを開始する。

【0063】PSCBエントリを処理するステップは次の通りである。

- ・PSCBエントリが空エントリである場合、これは、最初のNBTビットがキーと同一であるリーフがツリー内に無いことを意味するので、サーチは失敗する。
- ・PSCBにリーフのポインタがある場合には、PSC

Bからのポインタをリーフのアドレスと使用して、メモリからリーフが読み出される。リーフはレジスタに格納され、キーと比較される。このステップは、最後の比較と呼ばれる。完全一致が存在する場合、ツリー・サーチは成功する。それ以外では、ツリー・サーチは失敗する。

・PSCBがPSCBのポインタおよびNBTを持つ場合、NBTは最初に特定のレジスタに格納され、これは現在のNBTになる。次いでこのNBT数は、キー内のNBT位置のビットを見つけるために使用される。そのビット（0または1）をPSCBのポインタと共に使用して、正確な次のPSCBエントリが抽出される。そのビットはポインタの終わりに付加され、PSCBのメモリ内の完全なアドレスを提供する。PSCBは読み出され、特定のレジスタに格納される。次いでハードウェアは、PSCBエントリのこの処理を繰り返す。

【0064】ツリー・ウォーク中、リーフの全てのビットが試験されるのではなく、PSCB（ツリーの分岐）があるビットだけが試験される。したがって、リーフが見つかったら、リーフのパターンをキーと比較して、全てのビットが一致することを確認しなければならない。これが、このアルゴリズムの演算の最後の比較の理由である。サーチの成否は、完了フラッグと共にOK/KOフラッグによって印される。完了フラッグがトリガされると、このFMツリー・サーチ・エンジンを使用するプログラムまたはハードウェアは、OK/KOフラッグを調べることができる。

【0065】「プログラム可能」と記載されたものは全て、そのツリーに対応する特定のレジスタ値に設定することができる。エンジンがN個のツリーをサポートする必要がある場合には、これらの値がN個、レジスタ・アレイに入れられる。このレジスタには、プログラム可能な値、つまり使用するハッシュ関数、DTテーブルの開始、そのサイズ等が符号化される。

【0066】ハードウェアの1つの能力として、キーの自動挿入（ハードウェア挿入）がある。（ハッシュ化された）キーのサーチを進めながら、不一致（KO）があった場合、ハードウェアを使用して進行中にPSCBを形成することによって、そのポイントにリーフを自動的に挿入することができる。この場合、完全一致ツリー

の概念をキャッシュとして使用することができる。【0067】サーチは、直接テーブル108へのアクセスにより開始される。つまり、DTエントリは直接テーブル108から読み出される。DTエントリを読み出すために使用されるアドレスは、HashedKeyの最高位Nビットからだけでなく、ルックアップ定義テーブル（LUDefTable）で定義されるツリー特性に基づいても計算される。DTエントリは、ツリーのルートとみることができる。実際のツリー・データ構造は、ツリーの型に依存する。パトリシア・ツリー・データ構造はFMツリーに使

用され、パトリシア・ツリーの延長はLPMおよびSMTツリーに使用される。

【0068】8つのエントリのあるDT108の使用例を図6に示す。サーチ時間つまりアクセスしなければならないPSCBの数は、DT108を使用することによって減少できることが分かる。したがって、DTサイズを増加することによって、メモリの使用量とサーチ性能との間でトレードオフを行うことができる。

【0069】性能上の理由から、DTエントリを読み出して、そこにリーフのポインタが含まれることを見出すだけでは不十分であり、その後でリーフ自体を読み出さなければならない。この状況は、DTエントリ当たりの単一リーフ・エントリが多数存在するFMツリーの場合、非常にしばしば発生する。直接リーフの概念は、より多くのメモリの使用量とより優れた性能との間のトレードオフを可能にする。

【0070】ツリーは直接リーフをイネーブルすることができ、これはルックアップ定義テーブル（LUDefTable）で指定される。直接リーフをイネーブルしたツリーとディセーブルしたツリーとの相違を、図7に示す。直接リーフがイネーブルされ、DTエントリが単一リーフを含む場合、このリーフ130はDTエントリ自体に直接格納される。それ以外の場合、DTエントリはリーフのポインタを含む。

【0071】形状設定はツリー・サーチ・メモリ（TSM）の特徴であり、リーフまたはPSCBのようなオブジェクトをTSMにどのように格納するかを指定するために使用される。形状は幅および高さパラメータによって定義される。オブジェクトの高さは、オブジェクトが格納される連続アドレス場所の数を示す。オブジェクトの幅は、オブジェクトが格納される連続バンクの数を示す。幅および高さについて、ハードウェアは自動的に適切な数の場所を読み出す。ピココードの観点から、オブジェクトはアクセスの最小単位である。幅は、SRAMに格納されたオブジェクトの場合、常に1としなければならない。幅は、DRAMのオブジェクトの場合、2以上にすることができる。単一メモリ場所内に収まる十分に小さいオブジェクトは、1の高さおよび1の幅を持つように定義される。直接リーフをディセーブルしたDTエントリの形状は常に（W=1、H=1）である。DTエントリがダイナミック・ランダム・アクセス・メモリ（DRAM）に格納される場合、それはちょうど64ビットを占有する。直接リーフをイネーブルしたDTエントリの形状は、リーフの形状に等しく、それはLUDefTableに指定される。一般的に、これは、DT108によって使用されるメモリを増加させる。それはまた、DTエントリのアドレス計算に対するリーフの形状の影響を生じさせる。

【0072】DTエントリが読み出された後、DTエントリが直接リーフを含まず、空でもないとは仮定して、D

Tエントリから始まるツリーをウォークすることによって、サーチが続行される。ツリー・ウォークは、リーフに達するまで幾つかのPSCB（パターン・サーチ制御ブロック）を通過することができる。

【0073】FMツリーのサーチ中にPSCBに遭遇すると、ツリー・サーチ・エンジン・ハードウェア70は、HashedKeyのビットpの値によって、0分岐または1分岐にツリー・ウォークを続行する。

【0074】ツリー・ウォーク中、HashedKeyの全てのビットが試験されるのではなく、PSCBがあるビットだけが試験される。したがって、リーフが見つかったら、リーフのパターンをまだHashedKeyと比較して、全てのビットが一致することを確認しなければならない。リーフに格納されているのはHashedKeyであって、元の入力キーではないことに注意されたい。FMリーフが見つかったら、次の操作が実行される。

ステップ1：リーフ・パターンがHashedKeyと比較される。一致が発生すると、操作はステップ2に進む。そうでない場合、リーフが別のリーフへのチェーン・ポインタを含むならば、このリーフが読み出され、パターンが再びHashedKeyと比較される。一致せず、かつNLAフィールドが無ければ、サーチは失敗（KO）で終了する。

ステップ2：ベクトル・マスクがイネーブルされている場合、VectorIndex数を持つビットがリーフのベクトル・マスクから読み出される。このビットは、サーチ結果の一部として返される。サーチは成功（OK）で終了する。

【0075】図10は、本発明の完全一致サーチ・アルゴリズムの処理論理を示す。処理は、論理ブロック1000で入力キーの読出しから始まる。次いで入力キーに、論理ブロック1002に示すようにハッシュ関数が行われる。入力キーでハッシュ化を行なってハッシュ化キーを形成することは、任意選択である。ハッシュ関数は、ハッシュ化キーの最左端ビット、つまり直接テーブルをアドレス指定するために使用されるビットでエントロピーが最高となるように、選択される。ハッシュ関数は可逆である。つまり、ハッシュ化キーを入力キーに変換できる逆方向ハッシュ関数が存在する。次に、論理ブロック1004で、直接テーブルが読み出される。ハッシュ化キーの上位Nビット（それによりNは構成可能である）は、直接テーブルのインデックスとして使用される。読み出されたエントリが空である場合、サーチはKO（一致が見つからない）を返す。これは、終了ブロック1006によって示される。決定ブロック1008に示すように、エントリがリーフを指し示すか否かに関して決定が行われる。DTエントリがリーフを指し示す場合には、論理ブロック1010に示すように、そのリーフが読み出される。そうでない場合には、DTエントリはPSCBを指し示す。この場合、論理ブロック101

2に示すように、PSCBの適切な部分が読み出される。完全一致サーチの場合、PSCBは2つのエントリ、つまり0部分と1部分を含む。前PSCB（またはDTエントリ）は、ビット番号（NBT：試験する次のビット）を含む。NBTは、どのPSCBエントリを使用するかを選択するハッシュ化キー内のビット（つまり0または1）を選択する。PSCBエントリは、リーフのポインタまたは別のPSCBのポインタのいずれかを含む。次いで、処理は決定ブロック1008にループバックする。決定ブロック1008でリーフが見つかり、論理ブロック1010で読み出されると、論理ブロック1014で示すように、リーフに格納されたパターンが1ビットずつハッシュ化キーと比較される。決定ブロック1016に示すように、全てのビットが一致すると、サーチは、終了ブロック1018に示すように、OK（一致の成功）を返す。次いで、リーフの内容はアプリケーションに渡される。一致しない場合、終了ブロック1020に示すように、サーチはKO（失敗）を返す。この処理論理の延長として、ハッシュ化キーからのbビットでどのエントリをPSCBから読み出すかを選択するように、PSCBは2^b個のエントリで構成することができる。これにより、より多くのメモリ使用量を犠牲にして、性能が增強される。

【0076】ツリーのサーチ性能を增強するために、キャッシュを使用することができる。キャッシュの使用は、LUDefTableでツリー毎にイネーブルすることができる。サーチ中、ツリー・サーチ・エンジン70は最初にキャッシュを検査して、HashedKeyと一致するリーフが存在するかどうかを決定する。そのようなリーフが見つかる、それが返され、それ以上のサーチは必要ない。そのようなリーフが見つからなければ、通常のサーチが開始される。

【0077】ツリー・サーチ・エンジン・ハードウェア70にとって、キャッシュ・ルックアップは通常のサーチと全く同一である。したがって、入力キーはHashedKeyにハッシュ化され、直接テーブル108のアクセスが実行される。直接テーブル108はキャッシュとして作動する。キャッシュ・サーチがOK（成功）を返すと、サーチは終了する。そうでなければ、ツリー・サーチ・エンジン70は、ハッシュ演算が行われないことを除いて、全ツリーの第2サーチを開始する。HashedKeyレジスタ106の内容は再使用される。

【0078】キャッシュ・サーチを使用する場合、それはLUDefTableで指定することができる。キャッシュ・サーチがLUDefTableエントリIを使用し、サーチがKO（失敗）で終了すると、LUDefTableエントリI+1を使用して、別のサーチが自動的に開始される。原則的に、これにより複数のサーチを連鎖的に行うことができるが、LUDefTableのエントリI+1の下にツリー全体を格納することが推薦される。

【0079】ツリー・サーチ・エンジン70は、FMツリー、LPMツリー、およびSMTツリーでのハードウェア・サーチ作業を行う。三種類全ての場合に、ツリーを初期化し、維持するために変化する量のソフトウェアが必要になる。FMツリーおよびLPMツリーのみが、制御点プロセッサ34の介入無く、リーフの挿入および除去を実行できる能力を持つ。この機能の使用により、スケーラブルな構成が可能になり、かつ、必要な場合には、依然としてCP34がリーフを挿入または除去することができる柔軟性を備えている。

【0080】FMツリーは、テーブルで固定サイズ・パターンを効率的にサーチするための機構を提供する。これの一例は、レイヤ2イーサネット・ユニキャストMACテーブルである。イーサネット・ユニキャストMACアドレスは固定長の6バイトであり、厳密に一致しなければならない、さもなければ宛先は不明である。

【0081】FMツリーは、ハッシング関数から著しい利益が得られるので、最も性能の良いツリーである。ツリー・サーチ・エンジンは、衝突率が非常に低い複数の固定ハッシング関数を提供する。DT108が十分に大きいと仮定すると、複数のリーフが単一のDTエントリに関連付けられる確率は非常に低い。これは1+イプシロン・ルールであり、ここでイプシロンはDTエントリにおける衝突数を表す。1つのリーフを持つDTエントリは、イプシロン=0である。したがって、ハッシング関数を用いて、FMツリーを使用すると、イプシロンの値は非常に小さくなるはずである。

【0082】FMツリーのDTエントリの構造は、図8に見ることができる。各DTエントリは36ビット幅であり、次のフォーマットの1つを含む。

- ・空のDTエントリ。このDTエントリに関連付けられるリーフは無い。

- ・次のPSCBのポインタ。DTエントリは、PSCBのポインタを含む。次のPSCBアドレス(NPA)フィールドおよび次の試験ビット(NBT)フィールドが有効である。

- ・リーフのポインタ。DTエントリに関連付けられる単一のリーフがある。リーフ制御ブロック・アドレス(LCBA)は、このリーフのポインタを含む。

- ・直接リーフ。DTエントリに関連付けられる単一のリーフがあり、リーフはDTエントリ自体に格納される。リーフの第1フィールドはNLAロープでなければならない、これは直接リーフがループをイネーブルしなければならないことを暗示する。ループとは、ツリー内のリーフをひとつに連結するために使用される循環連結リストである。ピココードは「ロープをウォーク」して、ロープ内の全てのリーフを順次検査することができる。NLAの最初の2ビットは、自動的に「直接」と符号化するように、「10」を表すように予約されることに留意されたい。直接リーフは、これがUDefTableでイネーブル

された場合に、所与のツリーのためにだけ使用される。

- ・FM PSCBは、2つのPSCBラインで構成され、各PSCBラインが図8に示す2つのフォーマットのうちの1つを持つことができることを除いては、FM DTエントリと同一の構造を持つ。2つのPSCBラインはメモリ内で連続的に割り当てられ、ツリーをウォークするための分岐として使用される。次の試験ビット(NBT)フィールドは、PSCBをウォークするためのビット比較として使用するキーのオフセットを意味し、2つのPSCBラインのどちらを使用するかを表す。

【0083】7ビット値がツリーに格納される場合のFMツリーのサーチの一例を、図9に見ることができる。この例は、キーの最上位3ビット(MSB)をFM DT108へのハッシュとして使用することによって単純化される。このツリーに格納された5つのリーフ・エントリ(L0-L4)がある。

【0084】第1例として、1110011のバイナリ入力キーを想定する。最初の3ビット「111」はDTエントリ7に指標を付け、ここでリーフL0を指すLCBAが存在する。リーフL0はTSE70によって読み出され、L0のパターンが入力パターンと比較される。この例では、厳密な一致が発生し、TSEはOK(成功)を返す。

【0085】今、1001110の入力パターンを想定する。DTエントリ4は、NBTフィールドが3のPSCB0のポインタを含む。これは、キーの第4ビット「1」(ビット0はMSBつまり最上位ビット)がツリーのどの分岐を選ぶかを決定することを意味する。第4ビットは「1」なので、PSCB0の下半分が使用される。それが「0」であった場合、PSCB0の上半分が使用される。各PSCBは基本的にPSCBラインの2要素配列であり、ここで「0」のNBT値は第1要素を指標付けし、「1」のNBT値は第2要素を指標付けする。したがって、PSCB0のPSCBライン1は、6のNBTおよびPSCB2を指す次のPSCBアドレス(NPA)を含むので、サーチが続行される。NBTが7であり、入力パターンのビット7が「0」である場合、L3のポインタを含むPSCB2の上半分が使用される。リーフL3を読み出し、L3のパターンと入力パターンの完全比較作業を実行すると、OK(成功)が返される。

【0086】入力パターン1001100のサーチは、前の例の場合と全く同じツリー内の経路を辿るが、最後の比較作業は一致しないので、サーチはKO(失敗)を返す。

【0087】本発明は、ハードウェア、ソフトウェア、または2つの組合せで実現することができる。ここで説明する方法を実行するために適応させた任意の種類のコンピュータ・システムまたはその他の装置が適合する。

ハードウェアとソフトウェアの一般的な組合せは、ロードされ実行したときに、ここで説明する方法を実行するようにコンピュータ・システムを制御する、汎用コンピュータ・システムとすることができる。本発明はまた、ここで説明した方法の実現を可能にする全ての機能を含み、コンピュータ・システムにロードされたときにこれらの方法を実行することができる、コンピュータ・プログラム製品に埋め込むこともできる。

【0088】本発明の文脈におけるコンピュータ・プログラム命令またはコンピュータ・プログラムとは、情報処理能力を有するシステムに特定の関数をすぐに、またはa)別の言語、コードまたは表記法への変換、およびb)異なるマテリアル形式での複製、のいずれかまたは両方が行われた後で、実行させるように意図された1組の命令のいずれかの言語、コード(つまりピココード命令)、または表記法による表現を意味する。

【0089】当業者は、本発明の精神および範囲から逸脱することなく、本発明の好ましい実施形態に多くの変更が可能であることを理解されるであろう。さらに、他の機能を相応じて使用することなく、本発明の特徴の幾つかを使用することが可能である。したがって、本発明の範囲は請求の範囲によってのみ画定されるので、好ましい実施形態の上記の説明は、本発明の原理を解説することを目的として、本発明を制限することなく、提供するものである。

【0090】まとめとして、本発明の構成に関して以下の事項を開示する。

【0091】(1)入力キーをサーチ文字列として読み出す動作と、ハッシュ関数を使用して前記入力キーをハッシュ化してハッシュ化キーを生成する動作と、前記ハッシュ化キーの最上位Nビットを、各々の非空エントリがサーチ・ツリーの次の分岐またはリーフのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用する動作と、非空テーブル・エントリのポインタが、対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決定する動作と、ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐内容を読み出す動作と、対応するサーチ・ツリーのリーフに達した場合、リーフ内容を読み出し、リーフのパターンを前記ハッシュ化キーと比較して、前記リーフ・パターンが前記ハッシュ化キーと一致するかどうかを決定する動作と、前記リーフ・パターンが前記ハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返す動作とを含む、コンピュータ処理装置によって可変長サーチ・キーの完全一致を決定するための方法。

(2)前記サーチ・ツリーの複数のルート・ノードを表すテーブルが2^N個のエントリを含む、上記(1)に記載の完全一致を決定するための方法。

(3)前記コンピュータ処理装置がネットワーク・プロ

セッサである、上記(1)に記載の完全一致を決定するための方法。

(4)前記対応するサーチ・ツリーの次の分岐の内容が別の次の分岐を指し示す、上記(1)に記載の完全一致を決定するための方法。

(5)前記次の分岐の内容が対応するサーチ・ツリーのリーフを指し示す、上記(1)に記載の完全一致を決定するための方法。

(6)前記リーフ・パターンが前記ハッシュ化キーと一致せず、かつ別のリーフのポインタを含まない場合、一致見つからずの標識を返すことをさらに含む、上記

(1)に記載の完全一致を決定するための方法。

(7)テーブルのインデックスが空エントリのインデックスである場合、一致見つからずの標識を返すことをさらに含む、上記(1)に記載の完全一致を決定するための方法。

(8)色レジスタの内容を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、上記(1)に記載の完全一致を決定するための方法。

(9)ゼロの文字列を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、上記(1)に記載の完全一致を決定するための方法。

(10)前記次の分岐のビット数が前記ハッシュ化キーの長さを超える場合、完全一致のサーチを終了する動作をさらに含む、上記(1)に記載の完全一致を決定するための方法。

(11)前記入力キーに対して使用される前記ハッシュ関数が、前記ハッシュ化キーを前記入力キーに変換できる可逆ハッシュ関数である、上記(1)に記載の完全一致を決定するための方法。

(12)前記リーフが別のリーフへのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、前記格納されたパターンが前記ハッシュ化キーと一致せず、かつ前記チェーンの次のリーフのポインタを含まない場合、一致見つからずの標識を返す動作とをさらに含む、上記(1)に記載の完全一致を決定するための方法。

(13)前記リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、前記格納されたパターンが前記ハッシュ化キーと一致する場合、一致発見の標識を返す動作とをさらに含む、上記(1)に記載の完全一致を決定するための方法。

(14)サーチすべきパターンまたはキーと、サーチ・ツリーの第1アドレス位置を格納する直接テーブルと、各々がサーチ・ツリーの分岐を表す複数のパターン・サーチ制御ブロックと、各リーフがサーチの結果のためのアドレス位置である複数のリーフとを含む、可変長サー

チ・キーの完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(15) ツリー・サーチ・メモリを管理するルックアップ定義テーブルをさらに含む、上記(14)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(16) 前記ルックアップ定義テーブルは、ツリーが存在する物理的メモリ、キーおよびリーフのサイズ、および実行すべきサーチの種類を定義するエントリを含む、
10 上記(15)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(17) 前記ルックアップ定義テーブルが複数のメモリに実装される、上記(14)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(18) 直接テーブル・エントリのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、次の試験ビット、および直接リーフのうちの少なくとも1つを含む、
20 上記(14)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(19) パターン・サーチ制御ブロックのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、および次の試験ビットのうちの少なくとも1つを含む、上記

(14)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(20) リーフ・データ構造が1つのリーフ連鎖ポイント、プレフィックス長、サーチ・キーと比較すべきパターン、および可変ユーザ・データのうちの少なくとも1つを含む、上記(14)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(21) 前記直接リーフは直接テーブル・エントリに直接格納され、かつサーチ制御ブロックおよびサーチ・キーと比較されるパターンを含む、上記(18)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(22) パターン・サーチ制御ブロックが、サーチ・ツリーのリーフ・パターンが異なる位置に挿入される、
40 上記(14)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(23) パターン・サーチ制御ブロックは1の幅および1の高さによって定義される形状を持ち、かつ少なくとも36ビットのライン長を有するメモリに格納される、
上記(14)に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

(24) フレーム処理を行う複数のプロトコル・プロセ

ッサおよび内部制御点プロセッサを含む埋込み式プロセッサ複合体と、各プロトコル・プロセッサにアクセスでき、高速パターン・サーチ、データ操作、およびフレーム・パージングをもたらす複数のハードウェア・アクセラレータ・コプロセッサと、少なくとも1つのサーチ・ツリーを表す複数のデータ構造であって、直接テーブルとパターン・サーチ制御ブロックとリーフとを含む前記データ構造を格納する複数のプログラム可能なメモリ装置と、各プロトコル・プロセッサの複数のメモリ装置へのアクセスを制御する制御メモリ・アービタとを含む、
可変長サーチ・キーの完全一致を決定するために半導体基板上に組み立てられた装置。

(25) プロトコル・プロセッサの実行と並行して作動して、メモリの読み書きおよびメモリ範囲検査を含むツリー・サーチ命令を実行するツリー・サーチ・エンジンをさらに含む、上記(24)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(26) 前記複数のメモリ装置は内部スタティック・ランダム・アクセス・メモリ、外部スタティック・ランダム・アクセス・メモリ、および外部ダイナミック・ランダム・アクセス・メモリの少なくとも1つをさらに含む、
上記(24)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(27) 前記制御メモリ・アービタが、複数のプロトコル・プロセッサと複数のメモリ装置との間でメモリ・サイクルを割り当てることによって、制御メモリ動作を管理する、
上記(24)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(28) 各プロトコル・プロセッサは一次データ・バッファ、スクラッチ・パッド・データ・バッファ、およびデータ格納動作のための制御レジスタを含む、
上記(24)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(29) サーチ・キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに含む、
上記(24)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(30) プログラム可能なサーチ・キー・レジスタおよびプログラム可能なハッシュ化キー・レジスタをさらに含む、
40 上記(24)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(31) 複数の独立サーチ・ツリーの間で単一のテーブル・データ構造を共用することを可能にするプログラム可能な色キー・レジスタをさらに含む、
上記(30)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(32) 色レジスタがイネーブルされている場合、その内容をハッシュ出力に付加して最終ハッシュ化キーを生成する、
50 上記(31)に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(3 3) 色レジスタがイネーブルされていない場合、同等の数のゼロをハッシュ出力に付加して最終ハッシュ化キーを生成する、上記 (3 1) に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

(3 4) 入力キーをサーチ文字列として読み出すプログラム命令と、ハッシュ化キーを入力キーに変換することができる可逆ハッシュ関数を使用して前記入力キーをハッシュ化するプログラム命令と、前記ハッシュ化キーの最上位 N ビットを、各非空エントリがサーチ・ツリーの次の分岐またはリーフへのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用するプログラム命令と、非空テーブル・エントリのポインタが対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決定するプログラム命令と、ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐の内容を読み出すプログラム命令と、対応するサーチ・ツリーのリーフに達したときにリーフ内容を読み出し、リーフのパターンをハッシュ化キーと比較して、リーフ・パターンがハッシュ化キーと一致するかどうかを決定するプログラム命令と、リーフ・パターンがハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返すプログラム命令とを含む、可変長サーチ・キーの完全一致を決定するためのコンピュータ・プログラム製品を含むコンピュータ読み出し可能な媒体。

(3 5) サーチ・ツリーの複数のルート・ノードを表すテーブルが 2^N 個のエントリを含む、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(3 6) 前記コンピュータ処理装置がネットワーク・プロセッサである、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(3 7) 前記対応するサーチ・ツリーの次の分岐の内容が別の次の分岐を指し示す、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(3 8) 前記次の分岐の内容が対応するサーチ・ツリーのリーフを指し示す、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(3 9) リーフ・パターンがハッシュ化キーと一致せず、かつ別のリーフのポインタを含まない場合、一致見つけからずの標識を返すプログラム命令をさらに含む、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(4 0) テーブルのインデックスが空エントリのインデックスである場合、一致見つけからずの標識を返すプログラム命令をさらに含む、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(4 1) 色レジスタの内容をハッシュ化キーに付加して最終ハッシュ化キーを提供するプログラム命令をさらに含む、上記 (3 4) に記載の完全一致を決定するための

コンピュータ・プログラム製品。

(4 2) 文字列またはゼロをハッシュ化キーに付加して最終ハッシュ化キーを提供するプログラム命令をさらに含む、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(4 3) 次の分岐のビット数がハッシュ化キーの長さを超える場合、完全一致のサーチを終了するプログラム命令をさらに含む、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(4 4) リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンをハッシュ化キーと比較するプログラム命令と、格納されたパターンがハッシュ化キーと一致せず、かつチェーンの次のリーフのポインタを含まない場合、一致見つけからずの標識を返すプログラム命令とをさらに含む、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

(4 5) リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンをハッシュ化キーと比較するプログラム命令と、格納されたパターンがハッシュ化キーと一致する場合、一致発見の標識を返すプログラム命令とをさらに含む、上記 (3 4) に記載の完全一致を決定するためのコンピュータ・プログラム製品。

【図面の簡単な説明】

【図 1】本発明の好ましい実施形態によるネットワーク・プロセッサの例示的アーキテクチャを示すブロック図である。

【図 2】本発明の好ましい実施形態による埋込み型プロセッサ複合体の例示的実施形態を示すブロック図である。

【図 3】本発明の好ましい実施形態による例示的プロトコル・プロセッサ構造を示すブロック図である。

【図 4】本発明の好ましい実施形態による例示的イングレスおよびエグレス・フレームの流れを示すブロック図である。

【図 5】本発明の好ましい実施形態による完全マッチ・サーチ・アルゴリズムのためのツリー・データ構造を示す略図である。

【図 6】本発明の好ましい実施形態による直接テーブルを使用することの例示的データ構造への効果を示す略図である。

【図 7】本発明の好ましい実施形態による直接リーフを使用可能にすることの例示的データ構造への効果を示す略図である。

【図 8】本発明の好ましい実施形態による完全マッチ・サーチ・ツリーの D T エントリおよびパターン・サーチ制御ブロック (P S C B) ライン・フォーマットの例示的構造を示す表である。

【図 9】本発明の好ましい実施形態による完全マッチ・

サーチを使用するサーチの例を示す略図である。

【図10】本発明の好ましい実施形態による完全マッチ(FM)サーチ・アルゴリズムの処理論理を示す図である。

【図11】本発明の好ましい実施形態による例示的参照定義テーブルの内部構造を示す表である。

【図12】PSCBレジスタの内部フォーマットを示す表である。

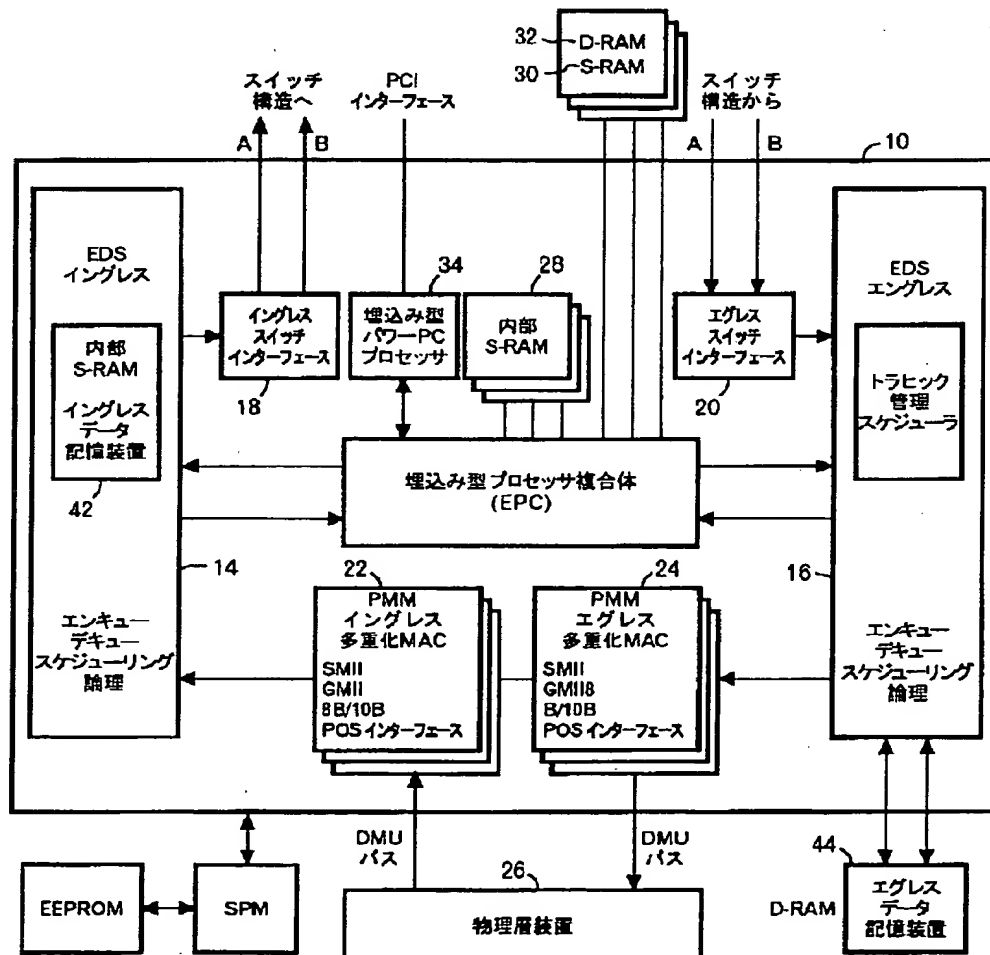
【図13】FMツリーの固定リーフ・フォーマットを示す表である。

【図14】本発明の好ましい実施形態によるツリー・サーチ・エンジンの例示的アーキテクチャを示すブロック図である。

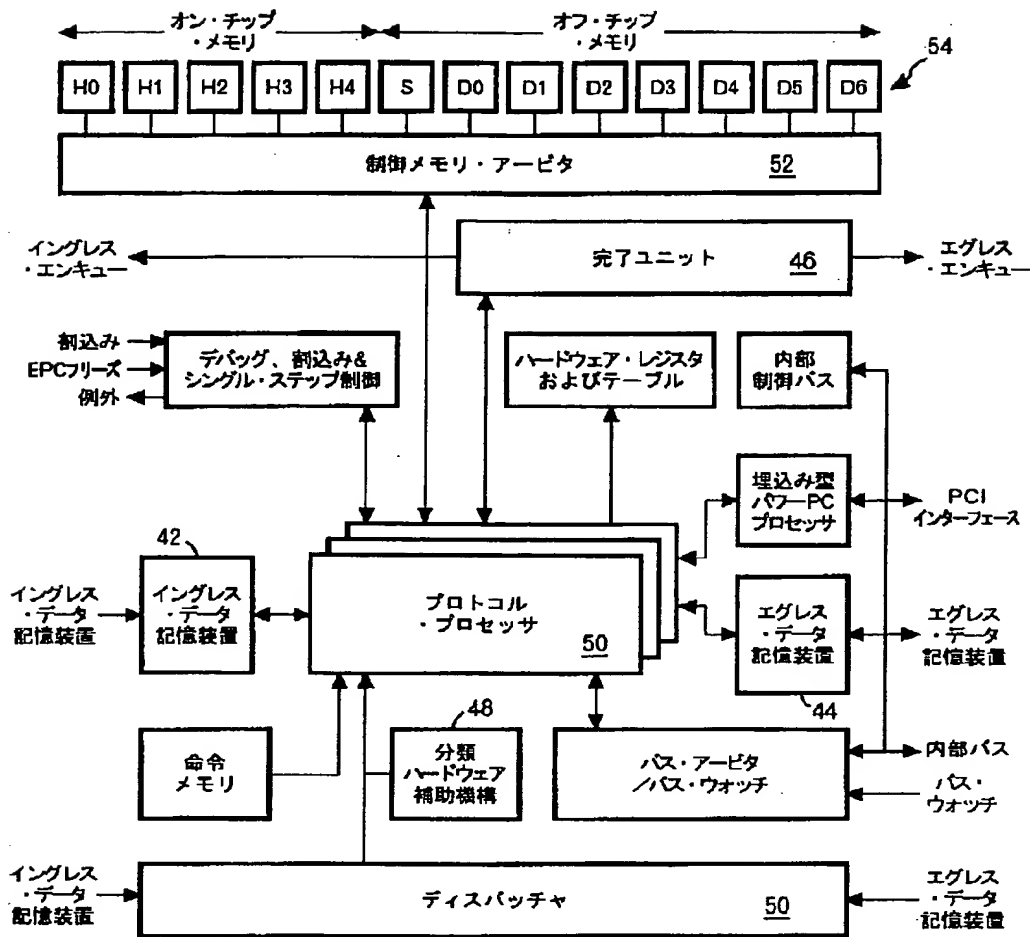
【符号の説明】

- 10 ネットワーク・プロセッサ
- 12 埋込み型プロセッサ複合体
- 14 エンキュー・デキュー・スケジューリング論理
- 18 イングレス・スイッチ・インタフェース
- 20 エグレス・スイッチ・インタフェース
- 28 スタティック・ランダム・アクセス・メモリ (SRAM)
- 30 ゼロ・バス・ターンアラウンドSRAM
- 32 ダイナミック・ランダム・アクセス・メモリ (DRAM)
- 34 接続点プロセッサ

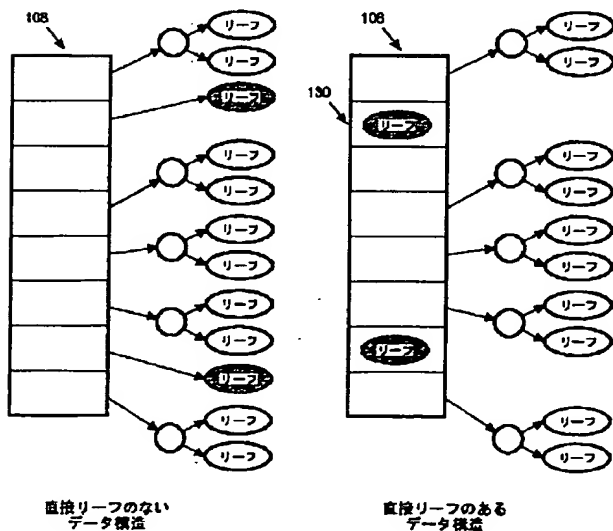
【図1】



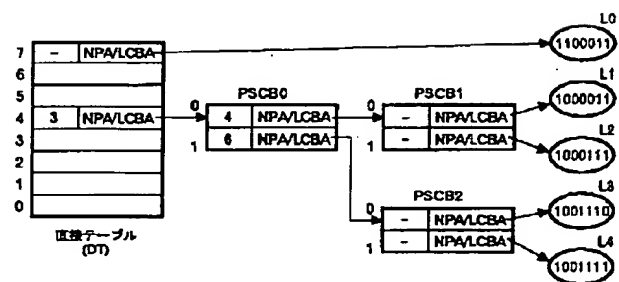
【図2】



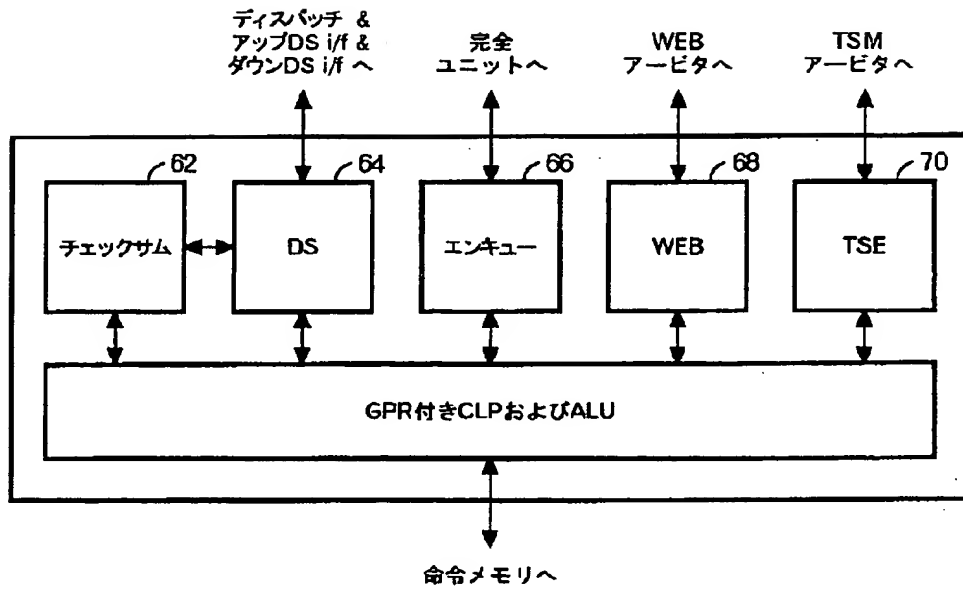
【図7】



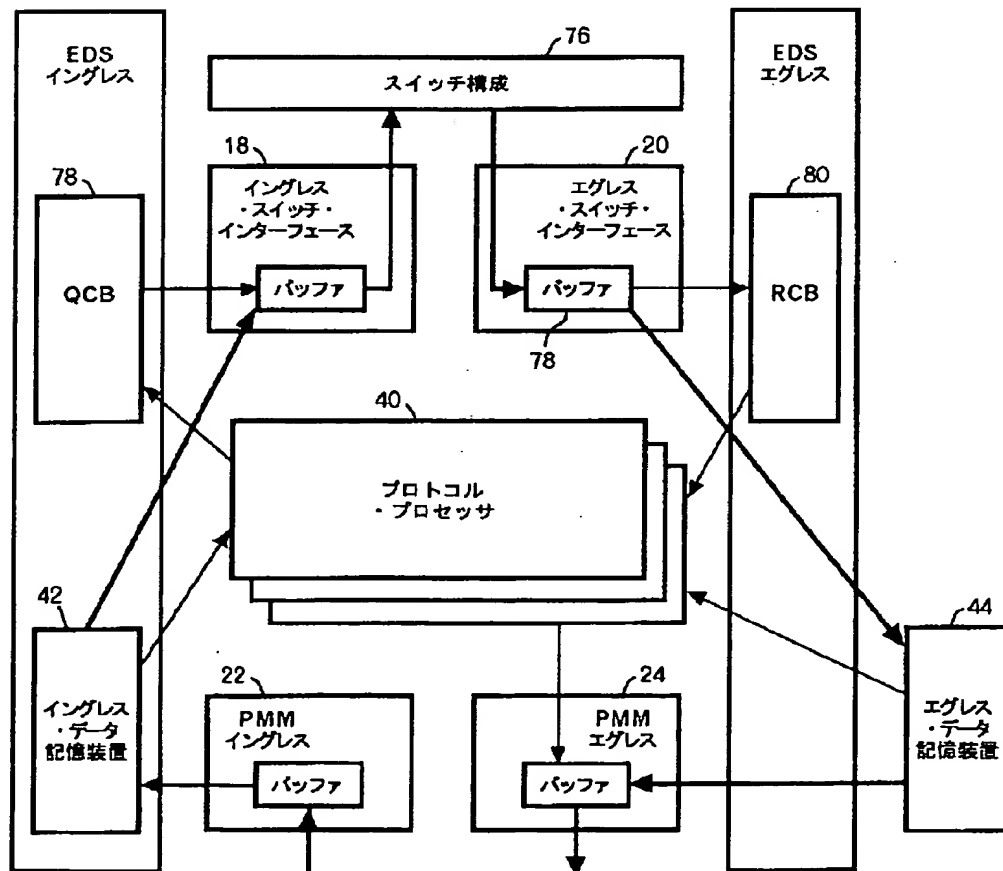
【図9】



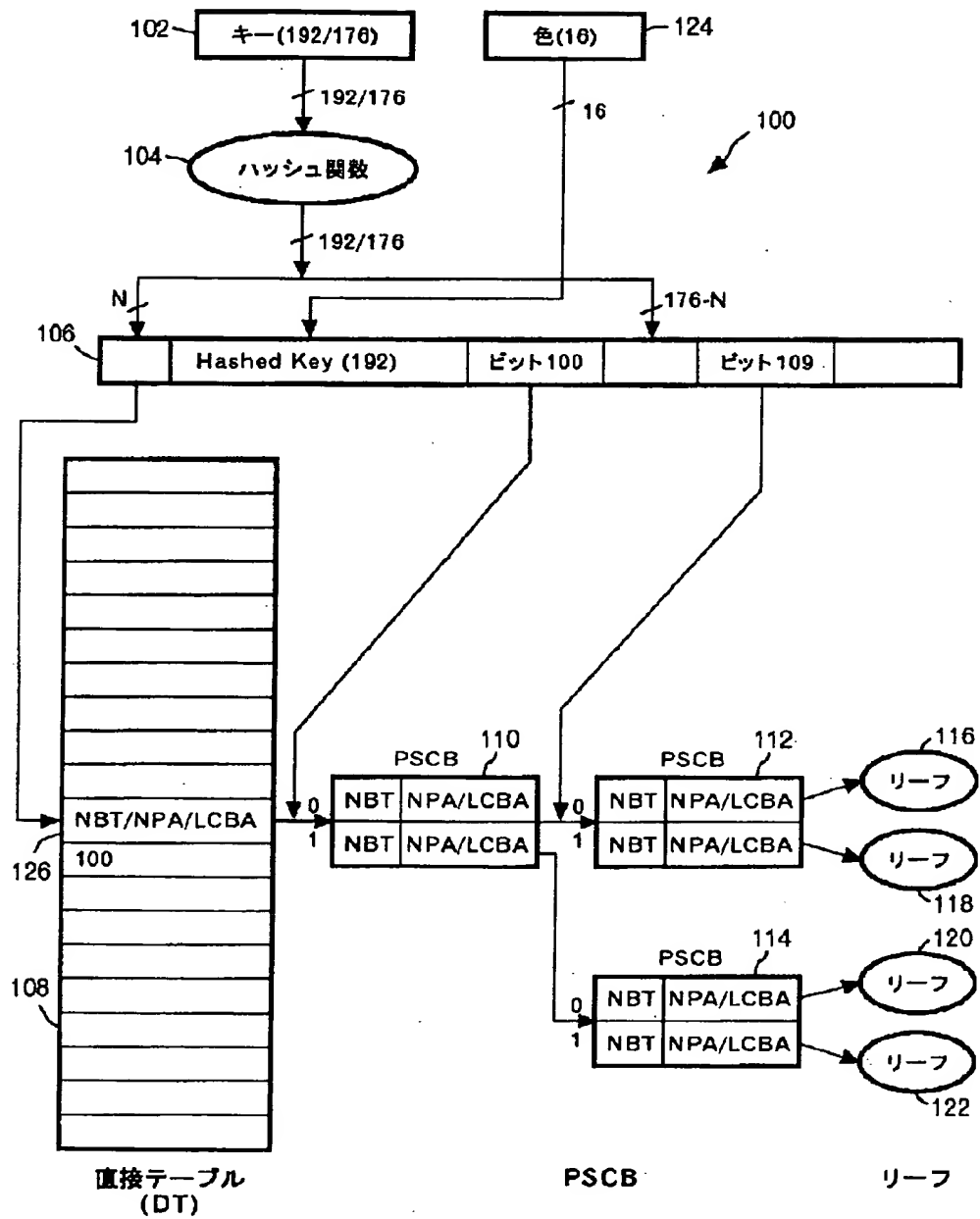
【図3】



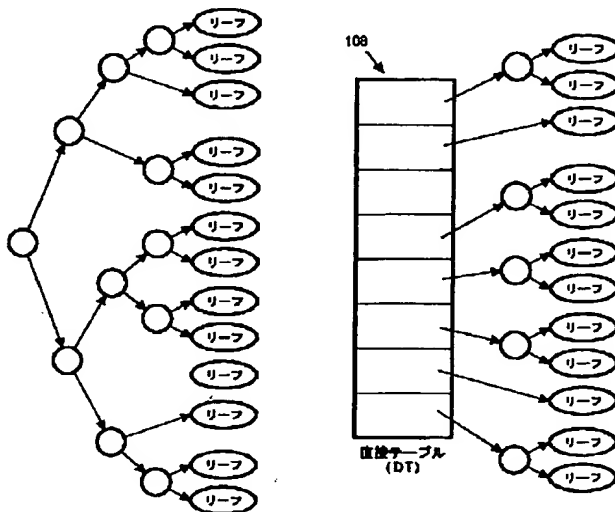
【図4】



【図5】



【図6】



直接テーブルを使用しない
データ構造

直接テーブルを使用する
データ構造

【図11】

LUDefTableの3つの定義

フィールド	サイズ	ビット
CacheEntry	1	0
Tree_Type	2	2..1
hash-type	4	6..3
color_en	1	7
PIP2_max_size	5	12..8
NPARope_en	1	13
NPASMT_en	1	14
CompIndex_en	1	15
PSCB_fq_index	6	21..16
PSCB_Height	1	22
Mask_Vector_En	1	23
CompIndex	8	31..24
DT_base_addr	26	57..32
DT_size	4	61..58
DT_interleaf	2	63..62
Leaf_fq_index	6	69..64
Leaf_Width	2	71..70
Leaf_Height	3	74..72
DirectLeafEn	1	75

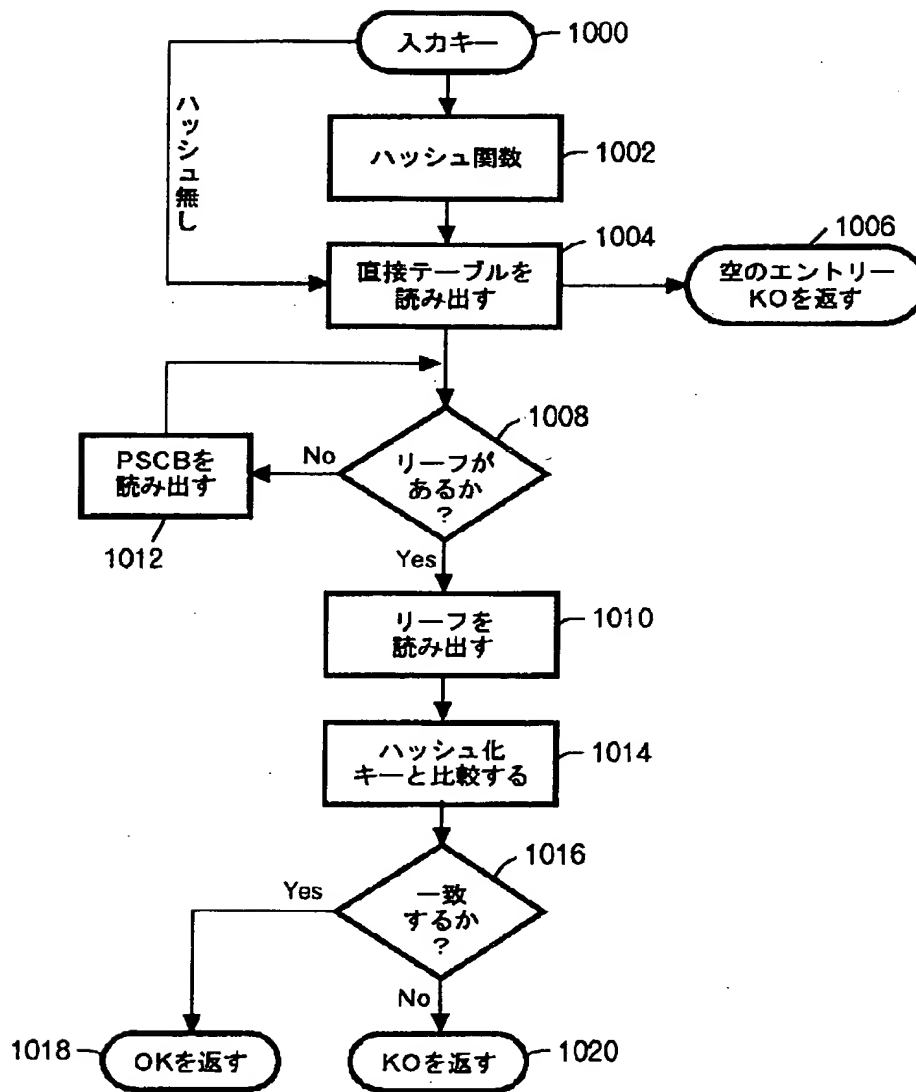
【図8】

フォーマット	状態	DTエン트리で有効か?	DSCBで有効か?	フォーマット (2ビット)	NPA/LCBA (26ビット)	NBT (8ビット)
空のDTエン트리	リーフ無し	有効	無効	00	0	0
次のPSCBのポインタ	DTエントリはポインタを含む	有効	有効	00	NPA	NBT
リーフのポインタ	単一のリーフがDTエントリに関連付けられる；LCBAフィールドはポインタを含む	有効	有効	01	LCBA	0

【図12】

フィールド	サイズ	PSCBが配置されるTSMのアドレス
NPA0	26	次のPSCBアドレス： PSCBの0部分のツリーの次のPSCBのポインタ
NBT0	8	PSCBの0部分の次の試験ビット
NCBA0	26	リーフ制御ブロックのアドレス： PSCBの0部分のリーフのポインタ
NPA1	26	次のPSCBアドレス： PSCBの1部分のツリーの次のPSCBのポインタ
NBT1	8	PSCBの1部分の次の試験ビット
LCBA1	26	リーフ制御ブロックのアドレス： PSCBの1部分のリーフのポインタ
Index	8	このPSCBのインデックス (物理的にPSCBに格納される)
PatBit	1	PSCBレジスタ内のインデックス・フィールドの値に基づくHashedkey(インデックス)の値

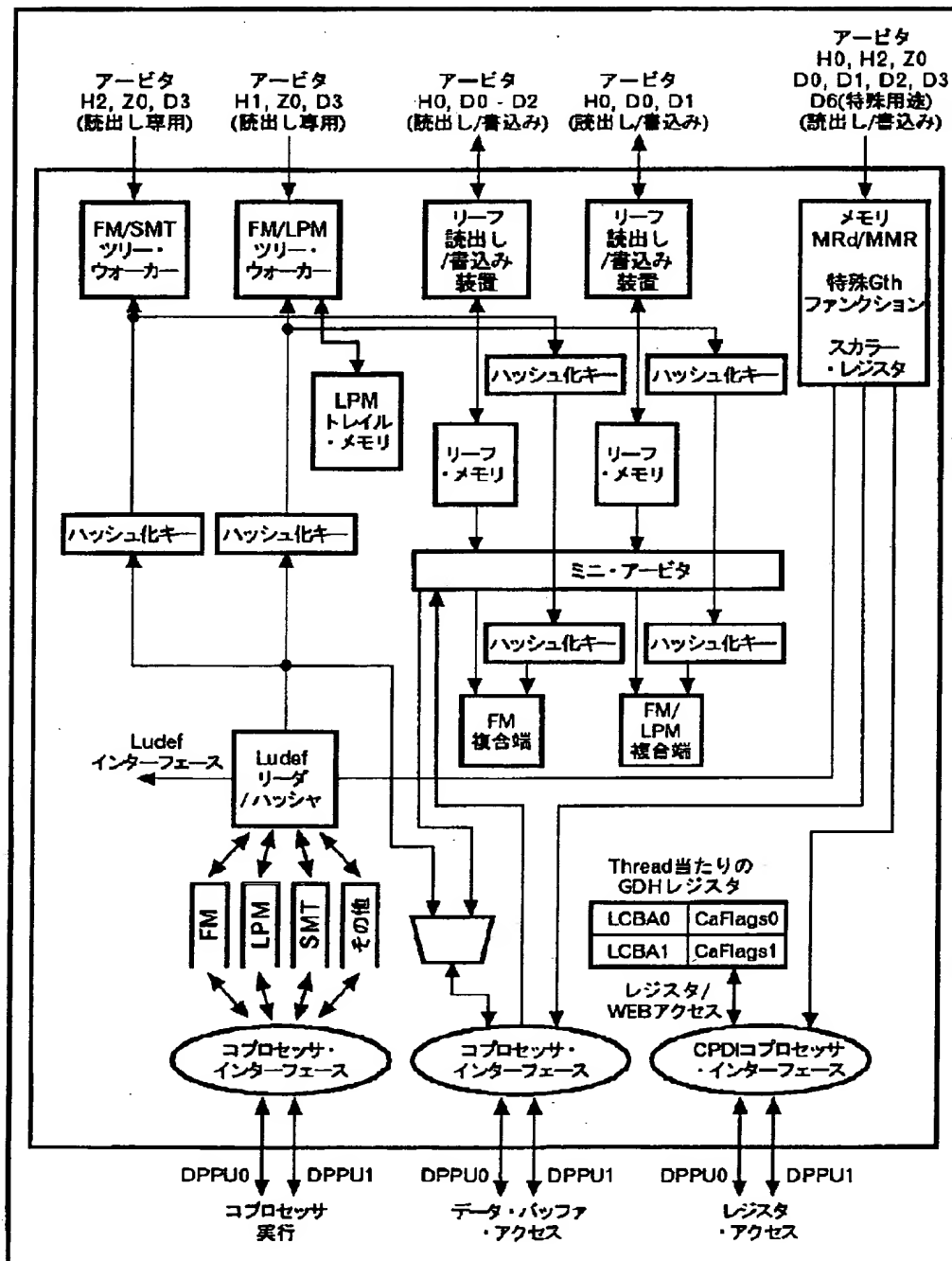
【図10】



【図13】

フィールド名	長さ	内 容
NLARope	4 バイト	リーフ連鎖ポインタ・エージング情報、および直接リーフ情報
Prefix_Len	1 バイト	このフィールドはFMツリーの場合TSFIによって使用されず、ピココードによって使用することができる
Pattern	2 - 18 バイト	Hashedkeyと比較されるパターン
UserData	可変長	このフィールドの内容は完全なピココードの制御下にある； UserDataフィールドは1つまたはそれ以上のカウンタを含むことができる

【図14】



フロントページの続き

(72)発明者 ブライアン・ミッチェル・バス
 アメリカ合衆国27502 ノースカロライナ
 州アベックス オールド・スターブリッ
 ジ・ドライブ4021

(72)発明者 ジャン・ルイ・カルヴィニャク
 アメリカ合衆国27511 ノースカロライナ
 州ケアリ スプリング・ホロー・レーン
 112

(72)発明者 マーコ・シー・ヘデス
アメリカ合衆国27612 ノースカロライナ
州ローリー グランド・マナー・コート
4109 ナンバー308

(72)発明者 アントニオス・マラグコス
アメリカ合衆国27612 ノースカロライナ
州ローリー プリンセトン・ミル・パーク
ウェイ3321 アpartment201

(72)発明者 マイケル・スティーブン・スィーゲル
アメリカ合衆国27615 ノースカロライナ
州ローリー ラウリー・ドライブ10625

(72)発明者 ファブリス・ジャン・ヴェルプランケン
フランス06610 ラゴード ルート・ド・
カーニュ9152